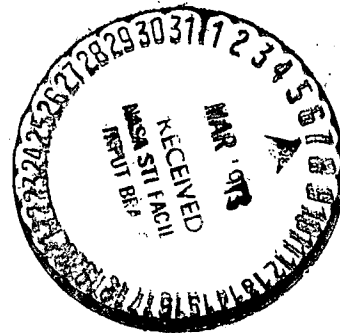


FINAL REPORT
Contract NAS1-11686

LOW THRUST ORBIT DETERMINATION PROGRAM

Prepared by:

P.E. Hong
G.L. Shults
K.R. Huling
C.W. Ratliff



Planetary Systems Mission Analysis and Operations Section
Martin Marietta Aerospace Division
Denver, Colorado

For

National Aeronautics Space Administration
Langley Research Center
Langley, Virginia

December, 1972

(NASA-CR-112256) LOW THRUST ORBIT
DETERMINATION PROGRAM Final Report
(Martin Marietta Corp.) 263 p HC \$15.25

CSCI 22A

G3/30 : 62817

Unclass

N73-17862

NASA CR-112256

FINAL REPORT

LOW THRUST ORBIT DETERMINATION PROGRAM

by P.E. Hong, G.L. Shults, K.R. Huling and C.W. Ratliff

Matrin Marietta Aerospace Division

Denver, Colorado

for Langley Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION o WASHINGTON, D.C. o DECEMBER 1972

ACKNOWLEDGEMENT

The authors wish to express their thanks for the invaluable technical support provided by Gentry Lee, the Program Manager. We also wish to thank Sandy Carlson and Suzy Borah for their effort in preparing flow charts, and Anne Chipman for her perseverance in typing our work of art.

TABLE OF CONTENTS

Acknowledgement	ii
Table of Contents	iii
Summary	1
1. Introduction	2
2. Nomenclature	4
3. Program Description (Functional Input/Output)	7
3.1 Targeting and Optimization Mode	9
3.2 Error Analysis Mode	15
3.3 Simulation Mode	22
4. Macrologic	26
4.1 Functional Flow	26
4.1.1 Targeting and Optimization (TOM)	29
4.1.2 Error Analysis (TEAM)	29
4.1.3 Simulation (TSIM)	32
4.2 Subroutine Hierarchy	36
5. Subroutine Descriptions	44
5.1 Subroutines Used in More Than One Mode	44
5.1.1 Utility Routines	44
5.1.1.1 CONIC	46
5.1.1.2 CONVRT	48
5.1.1.3 EULMX	49
5.1.1.4 PECEQ	50
5.1.2 Trajectory Routines	51
5.1.2.1 TRAJ	51
5.1.2.2 BPLANE	57
5.1.2.3 DERY	58

5.1.2.4	DETECT	60
5.1.2.5	ENCØN	61
5.1.2.6	EP	62
5.1.2.7	EPHEM	67
5.1.2.8	GRAVFØ	68
5.1.2.9	INTEG	71
5.1.2.10	PØWER	80
5.2	Targeting and Optimization Mode	81
5.2.1	BUCKET	81
5.2.2	DELU	84
5.2.3	FECS	87
5.2.4	FUNCT	90
5.2.5	GENMIN	93
5.2.6	GRID	99
5.2.7	MINMUM	102
5.2.8	SIZE	106
5.2.9	STEP	111
5.2.10	STEST	112
5.2.11	TEST	115
5.2.12	TØM	118
5.2.13	UPHILL	120
5.2.14	WEIGHT	123
5.3	Error Analysis Mode	125
5.3.1	CØVP	126
5.3.2	DATA	128
5.3.3	DYNØ	127
5.3.4	FILTER	133
5.3.5	GPRINT	137
5.3.6	GUIDM	138

5.3.7	KSGAIN	144
5.3.8	MEAS	145
5.3.9	MENØ	147
5.3.10	PATH	148
5.3.11	PRED	149
5.3.12	PRINT	150
5.3.13	PRØP	151
5.3.14	PTRAN	155
5.3.15	SCHED	159
5.3.16	SETEVN	161
5.3.17	STAPRL	162
5.3.18	STMGEN	164
5.3.19	STMRDR	166
5.3.20	TARPRL	168
5.3.21	TEAM	170
5.3.22	TRAKM	171
5.3.23	USRGAN	186
5.3.24	WLSGAN	187
5.3.25	XGUID	190
5.4	Simulation Mode	191
5.4.1	CSAMP	191
5.4.2	DATAS	193
5.4.3	GUIDS	195
5.4.4	NOISE	198
5.4.5	RNUM	199
5.4.6	SCØMP	200
5.4.7	SETUP	201
5.4.8	STAT	202
5.4.9	TARMAT	203

5.4.10 TSIM	205
6. Programming Guidelines	209
7. Future Options	211
8. References	214
9. Appendices	215
9.1 Error Sources	216
9.2 Covariance Accuracy	229
9.2.1 Preliminary Analysis	229
9.2.2 Final Results	238
9.3 PDØT vs. PHI	242

THE END

LOW THRUST ORBIT DETERMINATION PROGRAM

Prepared by:

P.E. Hong
G.L. Shults
K.R. Huling
C.W. Ratliff

SUMMARY

This document provides logical flow and guidelines for the construction of a low thrust orbit determination computer program. The program, tentatively called FRACAS (Filter Response Aalysis for Continuously Accelerating Spacecraft), is capable of generating a reference low thrust trajectory, performing a linear covariance analysis of guidance and navigation processes, and analyzing trajectory non-linearities in Monte Carlo fashion. The choice of trajectory, guidance and navigation models has been made after extensive literature surveys and investigation of previous software. A key part of program design relied upon experience gained in developing and using Martin Marietta Aerospace programs: TOPSEP (Targeting/Optimization for Solar Electric Propulsion), GODSEP (Guidance and Orbit Determination for SEP) and SIMSEP (Simulation of SEP).

1. INTRODUCTION

A major requirement for spacecraft systems design is an effective analysis of performance errors and their impact on mission success. This requirement is especially necessary for low thrust missions where thrust errors dominate all other error sources. Fast, accurate parametric error analyses can only be performed by a computer program which is efficiently constructed, easy to use, flexible, and contains modeling of all pertinent spacecraft and environmental processes. The FRACAS (Filter Response Analysis for Continuously Accelerating Spacecraft) program is designed to meet these characteristics. It is intended to provide rapid evaluation of guidance, navigation and performance requirements to the degree necessary for spacecraft and mission design.

This document describes the structure of FRACAS. The three basic program modes (trajectory generation, error analysis, simulation) are integrated in a master program which selects appropriate routines and performs the necessary executive control. The total primary and secondary overlay structure will require less than 70,000 octal words of a CDC 6000 series computer. Descriptions of the overall logic (macrologic) and of each major subroutine are contained in the following sections. To retain flexibility and growth potential, the program modules are designed with minimum interdependence.

Most of the technical and software experience used in designing FRACAS has been obtained from work with STEAP (Reference 1) and the low thrust programs TOPSEP (Targeting/Optimization), GODSEP (linear error analysis), and SIMSEP (trajectory simulation). Many other programs were used in a lesser, but still significant, degree; POST (Shuttle trajectory

optimization), SWEAT (Swingby error analysis), and BANANA (Bit Allocation Necessary for Accurate Navigation Analysis). All of these programs were developed by Martin Marietta Aerospace and have been applied in a variety of interplanetary mission analyses. Some of the major technical analyses which were performed to develop algorithms are summarized in the Appendices. The appendices are self-contained memoranda complete with their own references. The two most difficult technical problems were in determining 1) numerical accuracy of the covariance formulation and 2) method of covariance propagation including process noise model. These two problems were resolved satisfactorily and study results are summarized in Appendices 9.2 and 9.3, respectively.

2. NOMENCLATURE

The following symbols are used throughout the program and subroutine descriptions. However, deviations from these symbols may occur in localized discussion if required for purposes of clarity.

<u>SYMBOL</u>	<u>DEFINITION</u>
a	propulsive acceleration
c	propulsive exhaust velocity
C	cross covariance
E	target error index
F	net cost function
G	performance gradient
H	observation sensitivity matrix WRT state parameter
K	filter gain matrix
m	spacecraft mass
P	covariance
P_o	propulsive power at 1 AU
Q	dynamic noise matrix
Q	thrust noise matrix
r	spacecraft position
s	solve-for parameters
S	target sensitivity matrix WRT control parameters
t	arbitrary time
T	event time, .. target variables, or thrust
u	dynamic consider parameters
U	control parameters
U_o, V_o, W_o	a priori covariances on dynamic consider measurement consider and ignore parameters, respectively

SYMBOLDEFINITION

v	spacecraft velocity or measurement parameters
w	ignore parameters
x	spacecraft state
Γ	guidance matrix
η	propulsive efficiency or time-varying thrust error
θ	transition matrix of dynamic parameters
μ	gravitational constant
σ	standard deviation
τ	correlation time of thrust error
ϕ	transition matrix of augmented state
$\Phi(t_{k+1}, t_k)$	state transition matrix from time t_k to t_{k+1}
χ	target variation matrix

SUBSCRIPTDEFINITION

$()_A$	assumed covariance
$()_B$	true covariance
$()_C$	state control covariance
$()_{k+1,k}$	matrix evaluated over time interval t_k to t_{k+1}
$()_o, ()_k, ()_f$	evaluated at time t_o, t_k, t_f , respectively
$()_S$	solve-for parameter
$()_v$	measurement consider parameters
$()_w$	ignore parameters
$()_x$	spacecraft state parameters

SUBSCRIPTS $()_{xu}$

cross terms of state and dynamic consider
parameters

MISCELLANEOUS

OD

orbit determination

WRT

with respect to

 $E[\]$

expected value operation

 $()^+$

post-event value

 $()^-$

pre-event value

STM

State transition matrix

DEFINTIONDEFINTION

3. PROGRAM DESCRIPTION (FUNCTIONAL INPUT/OUTPUT)

FRACAS is a pre-mission design tool used for parametric studies of trajectory dispersions and their relationship with anticipated error sources. It is an intermediate step between early mission opportunity definition and precision real-time flight software. As such, the FRACAS design reflects a trade-off between computational speed vs. high numerical and modeling accuracy. The results of FRACAS are intended to provide 1) trajectory sensitivities to dynamic processes, 2) state estimation accuracies based upon an orbit determination (OD) algorithm and expected errors in the environment, spacecraft performance, and navigation system, 3) trajectory correction requirements in the form of ΔV and/or thrust control adjustments to return the trajectory to desired terminal conditions, and 4) probabilistic trajectory dispersions as a result of all significant dynamic, guidance and navigation processes.

FRACAS is divided into three modes which represent a logical sequence of analysis. The first mode generates a reference trajectory consistent with dynamic constraints. The user defines the mission in terms of launch and target planet, propulsion mode, flight time, etc. and then provides an estimate of desired control variables in the form of initial conditions and thrust parameters. The control parameters are varied within constraints such that the final trajectory meets all desired end conditions and maximizes spacecraft mass at the target. In addition to providing a reference mission for use in the next two program modes, information is available relating to trajectory sensitivities and non-linearities with respect to dynamic parameters.

The second FRACAS mode is a linear covariance analysis of the reference mission. Distributions of errors in the form of variances and covariances are applied in a probabilistic sense to the trajectory as a sequence of mission events is processed. The mission events include thrust switching, guidance correction and navigation measurement processing (OD). Usually, the time history of two types of trajectory errors are of interest: deviations of the actual trajectory from the reference, called control error, and deviations of the estimated trajectory from the actual, called knowledge error. Guidance events also provide probabilistic uncertainties in control corrections required to remove trajectory error at the event time.

A key assumption in the error analysis mode is linearity, that is, deviations about the reference trajectory behave in a linear fashion. The third FRACAS mode verifies this assumption, or at least defines regions of linearity. Discrete errors (randomly sampled from input statistical distributions) are applied to a deterministic trajectory. Guidance maneuvers are explicitly performed. By repeating the mission simulation with varying error samples, a Monte Carlo analysis can be constructed which takes into account the significant trajectory non-linearities. The simulation mode is of course the most lengthy in computer time and should be used primarily to support the error analysis mode.

Together, all three FRACAS modes provide the analyst with trajectory data necessary for proper spacecraft subsystem and mission design. The program is designed to be structurally simple and easy to use, yet maintain flexibility with respect to more sophisticated analysis by applying existing options or by program change.

Each mode will have its own namelist input although many of the variable names will be common to more than one namelist, e.g., basic spacecraft parameters. Printout options, punched output, and tape read/write will be controlled by namelist variables. At the beginning of every FRACAS data deck will be either an alphanumeric label or an integer which will determine the program mode.

3.1 Targeting and Optimization Mode

The targeting and optimization mode (TOM) generates a reference trajectory which is supplied as basic input to the error analysis and simulation modes. The primary purpose of TOM is to incorporate in this trajectory all of the desired flight characteristics for a particular interplanetary or near-Earth mission while optimizing the final spacecraft mass. Injection conditions, a thrusting time history, and other control parameters are found which accomplish this optimization and yet lead to the required target conditions. The target constraints may be the final spacecraft state (cartesian or B-plane coordinates), final orbital elements, radius of closest approach, or other mission specifications which are listed in the input entries later in this section.

Trajectories for the targeting and optimization mode are propagated using an Encke method with a two-step, fourth order Nystrom numerical integrator. Basically, the Encke method has been chosen to avoid integrating the entire vehicle acceleration vector to high precision. Since the accelerations due to a low thrust engine are considered small compared to the gravitational accelerations of the primary body, perturbation techniques can be applied. Integration is confined to evaluation of the relatively small deviations from the reference Encke conic resulting in rapid trajectory propagation. Conic propagation follows methods outlined in Battin (Reference 2).

The trajectory generation mode features a discrete parameter iteration algorithm which accommodates the nonlinear aspects of the low thrust problem. The algorithm is a modification of the POST Shuttle projected gradient method (Reference 3) and uses finite differencing techniques which compute performance and target sensitivities to control variations. These sensitivities direct the control selection to maximize the performance index while minimizing the target error index. The performance index is simply the value of the final spacecraft mass while the error index is the weighted sum of the squares of the target constraint errors.

The manipulation of trajectories to satisfy mission requirements is managed in the targeting and optimization submodes. TOM consists of four submodes which represent successive stages of trajectory development. These submodes are:

1. grid generation
2. trajectory targeting
3. a combination of trajectory targeting and optimization
4. trajectory optimization

Generally, these submodes are employed in order as listed above. However, any submode may be skipped or used individually if the proper control profile is available. For example, the linearity of controls characteristic of near-Earth missions, permits immediate entry into the targeting submode, although the control profile may not be extremely accurate. This is not the case of most interplanetary missions where nonlinearities require accurate control estimates to be input and the grid submode to be implemented. In situations where either an interplanetary or near-Earth mission is nearly targeted the third submode may be

employed initially. This submode provides control corrections for optimization of the trajectory in addition to completing the targeting. Finally, any trajectory which meets the targeting constraints can be optimized directly without entering the other submodes.

The grid generation submode is available to produce a number of trajectories which do not necessarily satisfy mission requirements but provide a range of trajectory solutions. Thus, the main purpose of the grid submode is to locate desirable control regions for further examination. In turn, each control is incremented a fixed amount while the remaining controls maintain their nominal values. A single low thrust trajectory is generated for each control change and the associated target error index is calculated. Then pairs of controls are incremented and the target error indices are computed from the resulting trajectories. Subsequently, contours of constant target error may be plotted in the control space so that some control regions can be eliminated from further consideration. Upon completion of the grid the trajectory generation mode is terminated and the program user must choose the best control profile to initialize targeting and optimization or to employ another grid approach.

When the targeting and optimization submodes are entered, a nominal trajectory is propagated directly from the input parameters. A series of tests is performed to determine which submode--targeting, optimization or both--is to be executed. If the target error index is large, the submode will be exclusively targeting. However, a target error index smaller than some arbitrary value (set in input) will result in simultaneous targeting and optimization. Whenever the index is below a specified lower bound, the optimization algorithm will be executed.

After the submode decision the basic projected gradient method is applied to the controls. The targeting sensitivity matrix S and performance gradient \underline{G} , are first computed. Elements of the S matrix represent the

sensitivities of individual target parameters to changes in controls and are used for both targeting and optimization. Similarly, the elements of the G vector represent the sensitivity of the performance index to changes in controls although these elements are used only for optimization. A weighting matrix which amplifies or diminishes the effects of the chosen controls is then calculated. Applying the projected gradient algorithm (Section 5.2), a new control vector direction is established. The magnitude of the control vector is determined by computing trial trajectories which adopt control profiles that lie in the new control vector direction. The new control profile is simply a scalar multiple of this control vector such that the targeting error index is minimized and/or the performance index is maximized. If the optimization is complete (the values of the performance index have converged to a maximum) TOM is terminated. Otherwise, the submode decision is made again and the cycle is repeated.

The speed of convergence for various missions depends largely on good control estimates. One method of computing control profiles and sizing system requirements for input into TOM is to apply the QUICKTOP program (Reference 4) developed by NASA/AMES to define low thrust interplanetary mission opportunities. QUICKTOP is approximate, self-starting, and computationally quick. The resulting values of the mission parameters can easily be adapted for refined targeting and optimization in the trajectory generation mode. Near-Earth missions, on the other hand, require less accurate control estimates. The input can usually be estimated by simple analytical calculations.

The targeting and optimization mode input is entered in the namelist

\$DATA and is read in the main subroutine TOM. A second namelist \$SMAG is used only when the targeting sensitivity matrix and performance gradient are to be input instead of calculated in the first iteration.

\$DATA

Nominal spacecraft parameters

- o initial mass
- o base power and power supply constants (e.g. nuclear decay rate)
- o thruster efficiency

Nominal thrust controls

- o thrust phase duration
- o pointing angles
- o thrust level
- o attitude mode

Trajectory and integration parameters

- o numerical integration accuracy level
- o initial spacecraft position and velocity
- o initial epoch
- o trajectory termination epoch
- o launch and target planets
- o array of codes of intermediate gravitational bodies to be considered
- o trajectory stopping conditions
 - oo sphere of influence
 - oo radius of closest approach
 - oo radius of designated final orbit

- o Control parameter codes chosen from the following list of available parameters:
 - initial spacecraft position and velocity,
 - exhaust velocity,
 - nominal thrust controls (e.g. thrust phase duration, pointing, thrust level, attitude modes)

Targeting parameters

- o desired target parameter values (any combination of the following parameters)
 - oo final spacecraft position and/or velocity
 - oo hyperbolic approach velocity
 - oo B-plane coordinates
 - oo time of arrival at sphere of influence
 - oo radius of closest approach to target body
 - oo time of **closest** approach
 - oo orbital elements
- o target tolerances

Submode parameters

- oo grid generation
- oo targeting and/or optimization
- oo nominal trajectory only
- o maximum number of iterations
- o number of control parameters
- o number of target conditions to be satisfied
- o maximum change allowed in performance in one iteration
- o limit of normalized targeting error below which "targeting only" is discontinued

- o percentage of targeting error to be corrected in first iteration
- o estimated radius of linearity region in control space
- o maximum value of control change scale factor
- o curve fitting tolerances for trial trajectories
- o control parameter perturbations
- o control parameter weightings
- o minimum angle between control vector elements in the control space below which an element is deleted from control profile

\$SMAG

- o targeting sensitivities
- o performance gradient

The trajectory information may be printed as a brief summary after each iteration or very detailed after each step of the projected gradient search. The detailed printout includes target sensitivities and weightings, performance gradients, trial trajectories, and control change scaling in addition to the desired spacecraft information throughout the optimized trajectory.

3.2 Error Analysis Mode

The error analysis mode performs a linear covariance analysis of guidance and navigation errors for low thrust trajectories. The underlying assumption is that all trajectory errors may be described as linear deviations from a reference trajectory, and that their ensemble statistics are Gaussian. Verification of this assumption for any trajectory may be made by exercising FRACAS simulation mode, described in Section 3.3 of this document.

Probabalistic a priori errors in the environment and spacecraft and tracking systems are propagated in time along the reference trajectory through

sequential events such as orbit determination (OD) and guidance corrections. Two types of ensemble error or covariances are distinguished - knowledge, which reflects the ability of the OD algorithm to estimate the spacecraft state; and control, which represents the dispersions of the actual spacecraft trajectory about the reference. Both knowledge and control covariances are stored internally in full covariance form rather than covariance square root form, the justification for which may be found in Appendix 9.2. Covariance propagation is done by either integration of covariance variational equations, or by the state transition matrix method. In general, the latter is recommended for reduced computer time (see Appendix 9.3).

Error analysis flow proceeds sequentially from start time to each specified trajectory event. Event types available are measurement, propagation, eigenvector, prediction, thrust on/off, and guidance. A measurement event processes tracking data at a time point by applying the user specified OD algorithm. Available to the user are both Kalman-Schmidt (K-S) and sequential weighted least squares (WLS) filters. The filters are distinguished by their methods of gain matrix calculation. FRACAS modularity also allows the user to insert his own filter algorithm quite easily.

A propagation event merely updates the knowledge covariance to the event time. Its primary value is in maintaining accurate covariance values during long propagations by forcing computation of the effective process noise over predetermined, user-specified intervals. Printout for the propagation event consists of the process noise covariance over the interval, which may be suppressed at the user's option.

An eigenvector event converts all covariance matrix sub-blocks to variable standard deviations and correlation coefficients, all of which are output. It also computes eigenvalues, their square roots, and eigenvectors for the position and velocity 3x3 sub-blocks of the state covariance matrix. Thrust on/off events are simply eigenvector events at the nominal thrust switching points.

A guidance event is an update of the control covariance to reflect implementation of a trajectory correction. A correction is not performed deterministically, but only in a probabilistic sense. Either impulsive ΔV or low thrust guidance can be performed (see Section 5.3.6). Low thrust guidance is further distinguished by being either primary or vernier. Vernier guidance is an update of a primary guidance correction to account for trajectory estimation improvement from tracking during the primary guidance interval. The guidance event computes, and displays to the user, expected correction covariances (Δv or thrust control), target error covariances before and after the guidance event, and the updated state control covariance.

The simplest form of the filtering algorithm available estimates the six-dimensional spacecraft state - three position and three velocity components. Since there are always additional parameters whose uncertainties are important to the OD process, the error analysis mode is designed to accommodate these. Parameters may be included in two categories - solve-for parameters, which are estimated simultaneously with the basic spacecraft state, and consider parameters, whose uncertainties are acknowledged by the filter, but which are not estimated. Consider parameters are divided into two types - measurement parameters which affect the measurement but not the dynamics, and dynamic parameters, those parameters which affect the dynamics and may or may not affect the measurements.

A major feature of the program is the inclusion of the generalized covariance option, a useful tool for studying filter sensitivity to mismodeling of real world error sources. When generalized covariance is exercised, two sets of knowledge covariances are operated on by the program. The first set, called assumed knowledge, comprises those covariances generated by the user selected filtering algorithm. The second set, called true knowledge, represents the effect the filtering algorithm has on true state estimation when real world error sources are not the same as those assumed by the filter. Mismatches between the two are effected either by setting true a priori uncertainties at different levels from assumed values, or the true state may be augmented by a vector of ignore parameters - parameters whose uncertainties are recognized by the true covariance analysis, but which are ignored by the assumed filter analysis. True covariance propagation and measurement updating is explained in Section 5.3.4, where the subroutine Filter is described.

The most significant time saving option available to the user is the creation of a state transition matrix (STM) file. Since many different studies are often made on the same reference trajectory, the user may specify an event schedule which will include all time points at which events may occur. The trajectory generation overlay will then generate the state transition matrices between these event times and store them on tape. During execution of the error analysis mode, this STM file is read to retrieve the necessary transition matrices. If event times exist on the STM file between any two events in a specific error analysis, the transition matrices are multiplied together to compute the total transition matrix over that time interval. The use of the STM file considerably

reduces integration time for multiple studies of a single reference trajectory. For maximum efficiency in this multiple study usage, the generation of the STM file must include transition matrix entries for all parameters which the user may at some time wish to solve-for, consider, or ignore. When the error analysis recovers these matrices from the STM file, entries corresponding to current parameters are loaded into the proper transition matrix partitions, and those for unused parameters are passed over.

Input to the FRACAS error analysis work is by namelists and event schedules where necessary. The first namelist ERRCON includes flags to indicate 1) if an STM file is to be created; 2) if the current run is supposed to execute an error analysis; and 3) if an error analysis is executed, whether the covariance propagation is to be by transition matrices or integration of covariance variational equations. If either an STM file is to be created, or the covariance variational equations option is selected, the namelist ERTRAJ is required which includes all input needed for reference trajectory generation by either method. Since trajectory integration is required for prediction and guidance events, even when an existing STM file is used, all of the information in namelist ERTRAJ is written at the beginning of the STM file and is read from that file rather than cards. This guarantees consistency of integration accuracy level, gravitating bodies used, and nominal spacecraft control policy between the STM file and these event integrations.

Immediately following ERTRAJ is a set of event scheduling cards defining all time points which must be written on the STM file. These cards are unnecessary if ERTRAJ is being read to initialize integration of covariance variational equations.

Next comes namelist ERANAL - which contains the basic information necessary for error analysis - followed by schedule cards for measurements and propagation events. Last, if generalized covariance is to be used, comes namelist GENCOV, which initializes relevant parameters. Inputs to GENCOV are minimized by assuming that all true covariance information is the same as that for the assumed filter analysis unless changed by namelist GENCOV.

Following are the error analysis mode namelists, and the input available through each;

Namelist \$ERRCON

- o STM file creation - true or false
- o Error analysis execution - true or false
- o Integration of covariance variational equations - true or false

Namelist \$ERTRAJ

- o Initial spacecraft state and flag indicating coordinate system
- o Spacecraft mass, exhaust velocity, thruster efficiency
- o Flag indicating power source
- o Base power and power system constants, e.g. decay rate of nuclear power
- o Initial date
- o Final date or total flight time
- o Gravitating bodies to be used for trajectory generation
- o Integration accuracy level
- o Target body
- o Ephemeris of target body if not available internally
- o Parameter list for state transition matrices
- o Control array defining thrust on/off times, and nominal control policies for thrusting arcs

Namelist \$ERANAL

- o All knowledge covariances describing augmented state
- o Parameter lists - solve-for, consider, ignore
- o Time varying thrust parameters, their uncertainties and their correlation times
- o Filtering algorithm flag (K-S or WLS)
- o Control covariances
- o Print flags
 - oo Print measurements according to time
 - oo Print measurements according to type
 - o Print measurements according to number, e.g.
 - every 12th measurement
 - oo Type of propagation event print
- o Number of measurement schedule cards to follow
- o Measurement noise levels
- o Station locations if additional or different stations from standard ones are desired
- o Event information
 - oo Number of propagation event cards to follow namelist
 - oo Number each of eigenvector, prediction and guidance events
 - oo Event timing information
 - oo Guidance policies and control weighting factors for each maneuver
- o Punch flags
 - oo Knowledge and/or control punched at specified times to initialize later error analyses or for input to simulation mode
 - oo Guidance variation matrices to eliminate recomputation in future error analyses

- o Generalized covariance flag - true or false

Namelist \$GENCOV

- o Ignore parameter list
- o A priori ignore parameter covariance terms
- o A priori true covariance terms which differ from corresponding assumed terms
- o True time varying thrust parameter information which differs from assumed
- o True measurement noise levels if different from assumed

3.3 Simulation Mode

The purpose of the simulation mode is to examine trajectory non-linearities as they affect final target errors. Discrete a priori errors in the environment and spacecraft systems are applied as the trajectory simulation proceeds through each scheduled guidance and navigation event. The form of the simulation mode is such that many missions can be simulated quickly, each with varying samples of error sources, from which a Monte Carlo error analysis can be constructed. Tracking is simulated by sampling an estimation error covariance prior to each guidance event. Estimation error or knowledge covariances would be obtained from the results of linear error analysis. The sampled state error is added to the current actual state to form a best estimate which is used to design the maneuver. There are many options which can be selected for maneuver design, namely, choice of target variables, conditions and tolerances, linear or nonlinear (iterative) guidance, impulsive or low thrust corrections, thrust control parameter weighting and constraints, etc. After the maneuver is designed, execution takes place by applying the design maneuver plus execution errors to the actual trajectory. When all guidance events have been completed the actual trajectory is propagated to the target and end

conditions are evaluated. Statistical error characteristics for desired parameters are constructed after each trajectory simulation.

The sampling of estimation error covariances, as opposed to explicit orbit determination of the actual trajectory, was chosen because it was computationally faster, enabling a Monte Carlo error analysis to be a reasonable undertaking. We felt that Monte Carlo analysis provided much more information than a single trajectory simulation. The Monte Carlo approach also provides the flexibility of taking an "interesting" trajectory from the set of simulated missions and using it as a reference trajectory for linear error analysis.

Because of the long run time necessary for a statistically significant Monte Carlo analysis, it is wise to break up the simulations into batches and keep the number of mission cycles per run to a minimum. Thus, capability exists in each FRACAS/simulation mode run to use the constructed error statistics of a previous run as a priori input and to punch cards containing cumulative statistics after the current run.

Simulation mode input is divided into two namelists. The first (\$INSIM) is for describing the reference mission and associated errors. The second namelist (\$INMAN) contains parameters describing a guidance correction event. Thus, each maneuver must have its own \$INMAN.

Namelist \$INSIM

- o nominal spacecraft parameters; exhaust velocity, available thruster power, thruster efficiency, initial mass
- o variances in spacecraft parameters
- o nominal initial spacecraft state
- o state error covariance

- o initial epoch
- o nominal thrust controls: phase duration, pointing, thrust level, attitude mode
- o thrust control variances (bias)
- o time-varying thrust errors: mean and variance of correlation time, variance in thrust direction and proportionality
- o execution error variances for impulsive maneuvers
- o covariance of planetary ephemeris errors (including gravitational constants)
- o launch planet, target planet, all other bodies to be considered
- o numerical integration accuracy level
- o random number initializer
- o print and punch flags
- o maximum number of mission cycles
- o number of maneuvers
- o number of mission cycles used to generate a priori error statistics
- o cumulative a priori error statistics (from previous runs)

Namelist \$INMAN:

- o maneuver epoch
- o estimation error covariance
- o guidance law: linear or non-linear, ΔV or low thrust
- o guidance policy: cutoff condition (time, sphere-of-influence, closest approach, radius) and target set (B-plane coordinates, cartesian, conic, Earth synchronous)
- o target body

- o target tolerances
- o thrust control tolerances and constraints
- o number of cycles used to generate a priori error statistics
- o cumulative a priori error statistics (from previous runs)
- o target sensitivity or guidance matrix, target conditions, nominal spacecraft state and mass at maneuver epoch are all optional input

Printout from the simulation mode can be a brief summary after each mission cycle or very detailed after each maneuver of each cycle. Cumulative statistics are always printed out at the end of the run and punched cards are available if desired. Some of the quantities displayed in the detailed printout will be deviations of actual parameters from their nominal values, cumulative means and standard deviations, target sensitivities and control steps after each iteration (for non-linear guidance), and reconstructed control and knowledge covariances.

4. MACROLOGIC

4.1 Functional Flow

The FRACAS program is a modular pre-flight analysis tool capable of generating targeted reference trajectories and performing error analyses on these trajectories as well as Monte Carlo trajectory simulations. FRACAS consists of three independent modules (TØM, TEAM and TSIM) illustrated in Figure 1. Each module performs the processing for its respective mode. FRACAS and the three modules are organized into an overlay structure to meet LRC imposed constraint of 70000₈ computer words on CDC 6000 series computers. The program FRACAS is a main overlay while TØM, TEAM and TSIM are primary overlays. Because of extensive computational functions, TEAM is the only module which requires secondary overlays, four in particular. Estimated core requirements for the entire overlay structure are illustrated in Figure 2. All estimates are based upon experience with MMA low thrust programs which perform functions similar to the proposed program. Should the estimates in Figure 2 be too optimistic, new secondary overlays could be created from the primary with subsequent increase in computer run time due to overlay loading. The total core requirement is estimated at about 67000₈ words.

A second LRC constraint of not more than 12 files has been met. FRACAS uses only four files: INPUT, ØUTPUT, PUNCH and a file (STM) used only in the error analysis module.

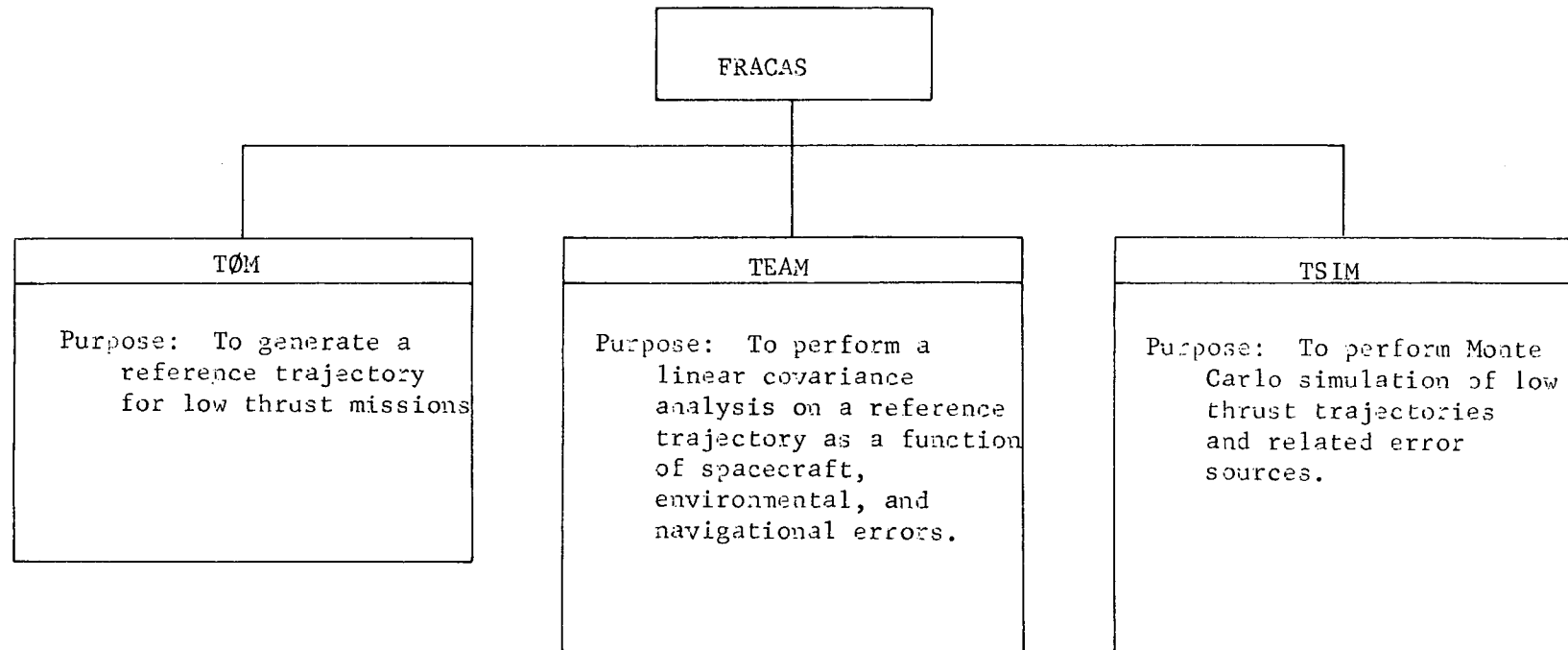


FIGURE 1. FRACAS Mode Structure

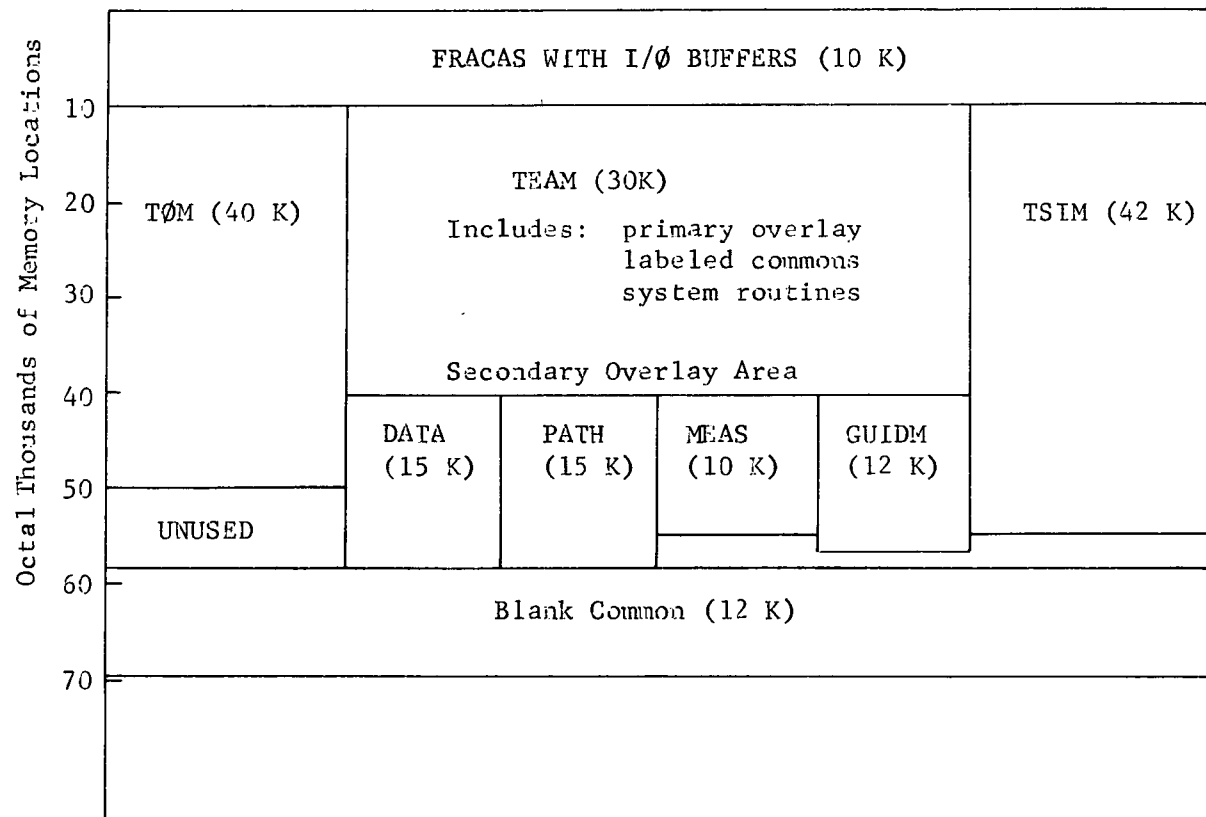


FIGURE 2. FRACAS Memory Map

4.1.1 Targeting and Optimization Module (TOM)

TOM generates trajectories which satisfy specified target and control parameter constraints and maximizes the final spacecraft mass at the target planet. The module has the ability to operate in four independent sub-modes:

- o grid generation - finds desirable control regions
- o trajectory targeting and optimization - generates optimized and targeted trajectory
- o trajectory optimization - optimizes a targeted trajectory

Generally, the submodes would be used in this sequence, however any sub-mode may be used if the proper control profile is available.

The functional flow of TOM is illustrated in Figure 3.

In the grid generation sub-mode a low-thrust trajectory is generated for each control change (initial conditions and thrust parameters) and the associated target error index is computed. Then pairs of controls are changed and the target error indices are computed. Contours of constant error may be computed in the control space so that sections of the control region can be singled out for further study.

In the targeting/optimization sub-modes a reference trajectory is generated or input which satisfies the control constraints and target conditions. If optimization is desired, changes are made to the controls to minimize a cost index. When a local minimum is found the optimization is completed. The projected gradient method is used for targeting and/or optimization. The targeting and optimization module is a single primary overlay of FRACAS.

4.1.2 Error Analysis Module (TEAM)

The error analysis module is used to examine trajectory dispersions resulting from thrusting, ephemeris, gravitational and measurement errors.

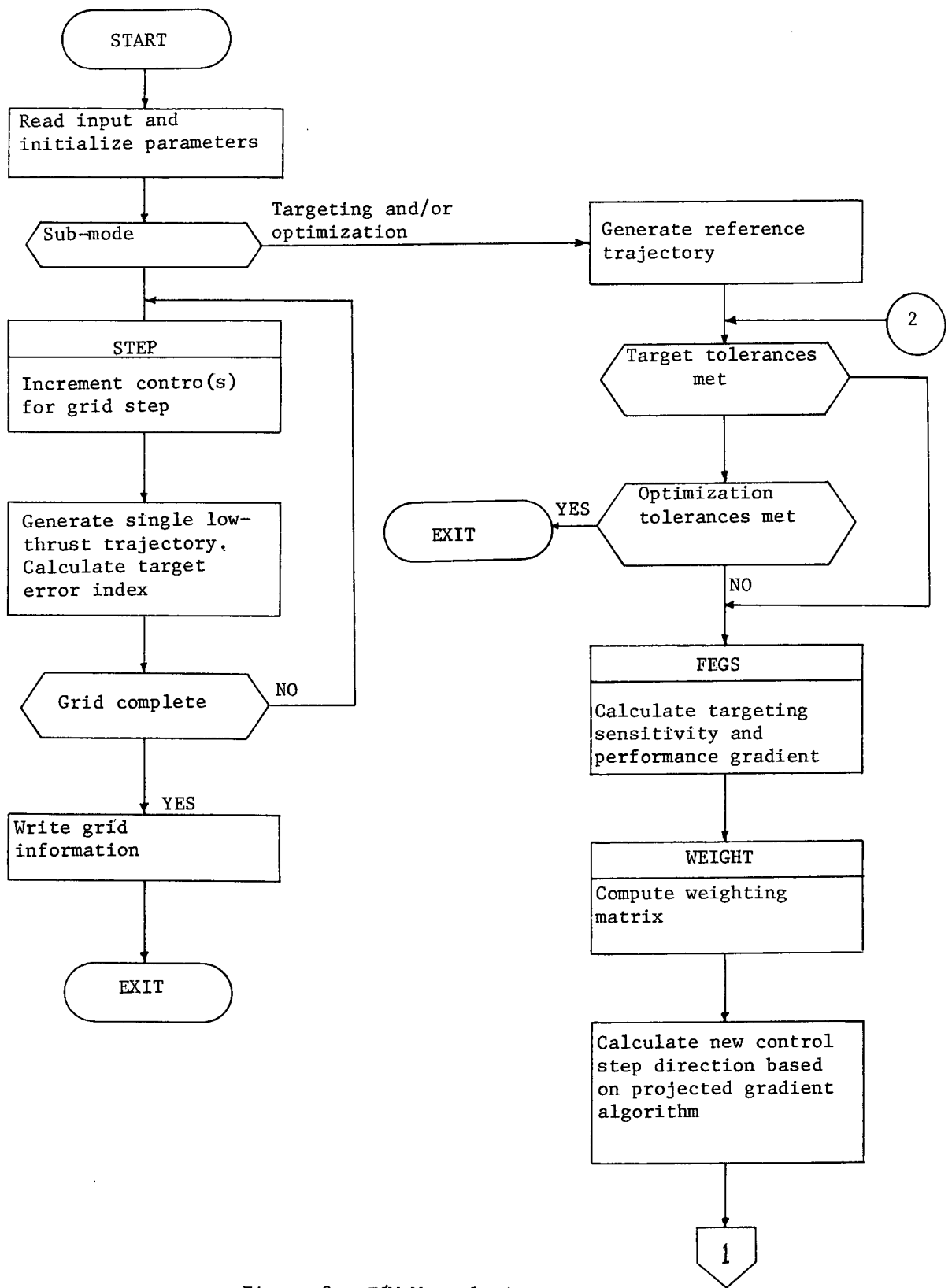


Figure 3. TOM Macrologic

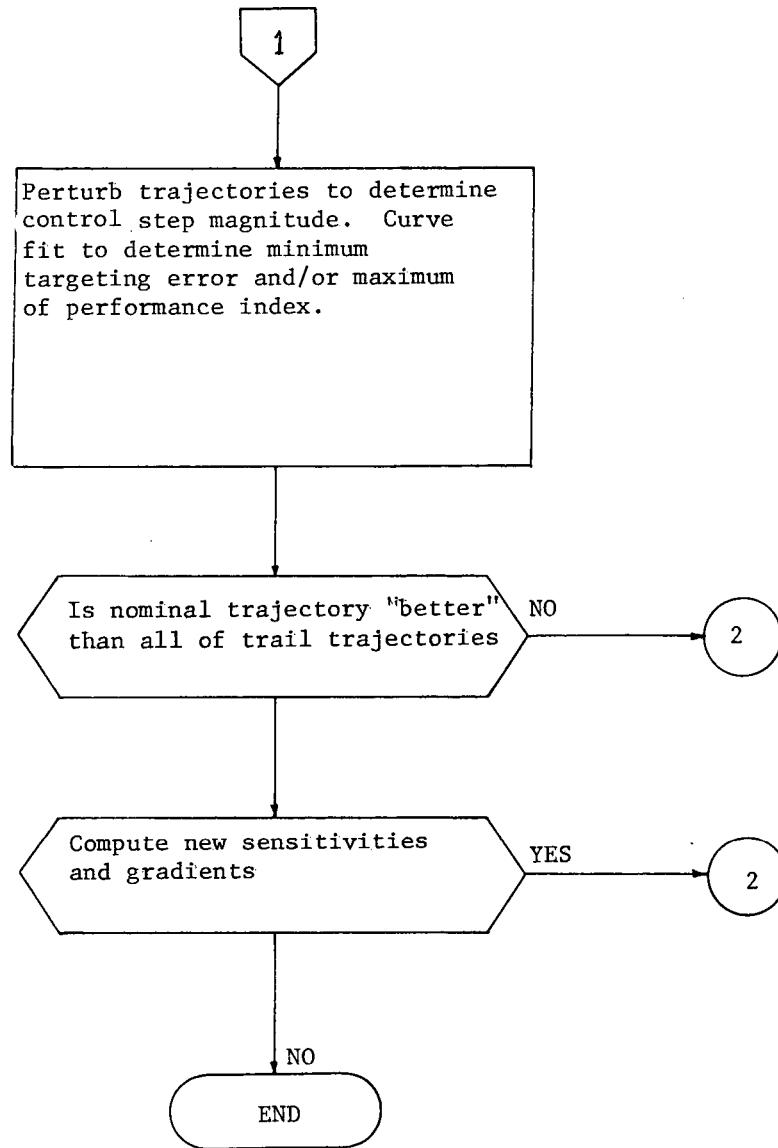


Figure 3. TOM Macrologic (continued)

Errors are represented as covariances which are propagated by state transition matrices. Error covariances are updated by either a Kalman-Schmidt or sequential weighted least squares filter. The module is broken down into a primary overlay and four secondary overlays. The primary overlay contains all logic necessary to control initialization and cycling of the error analysis mode. The secondary overlays are defined as follows:

1. DATA is responsible for all user input and editing; DATA will also do any initialization necessary for the proper functioning of the program.
2. PATH generates the state vector and mass of the spacecraft and the transition matrices from the previous to the current time.
3. MEAS processes measurements by computing measurement noise and observation matrices, and updating the state covariance using the recursive estimation algorithm.
4. GUIDM performs guidance events.

Logic flow is shown in Figure 4. DATA is called to read the user's input and check for inconsistencies and omissions. DATA also performs some initialization such as zeroing variables and setting up event scheduling. For multiple runs using the same reference trajectory, the user can create a file (STM) containing the integrated state and transition matrices at each event time. On successive runs, the information from the file can be used instead of integrating the same trajectory repeatedly. The basic cycle consists of obtaining the time of the next event, propagating the covariances to that time and calling the appropriate overlay to process the event, completing the cycle.

4.1.3 Trajectory Simulation Module (TSIM)

TSIM is used to examine the regions of linearity for low thrust trajectories as they affect target dispersions and thrust guidance

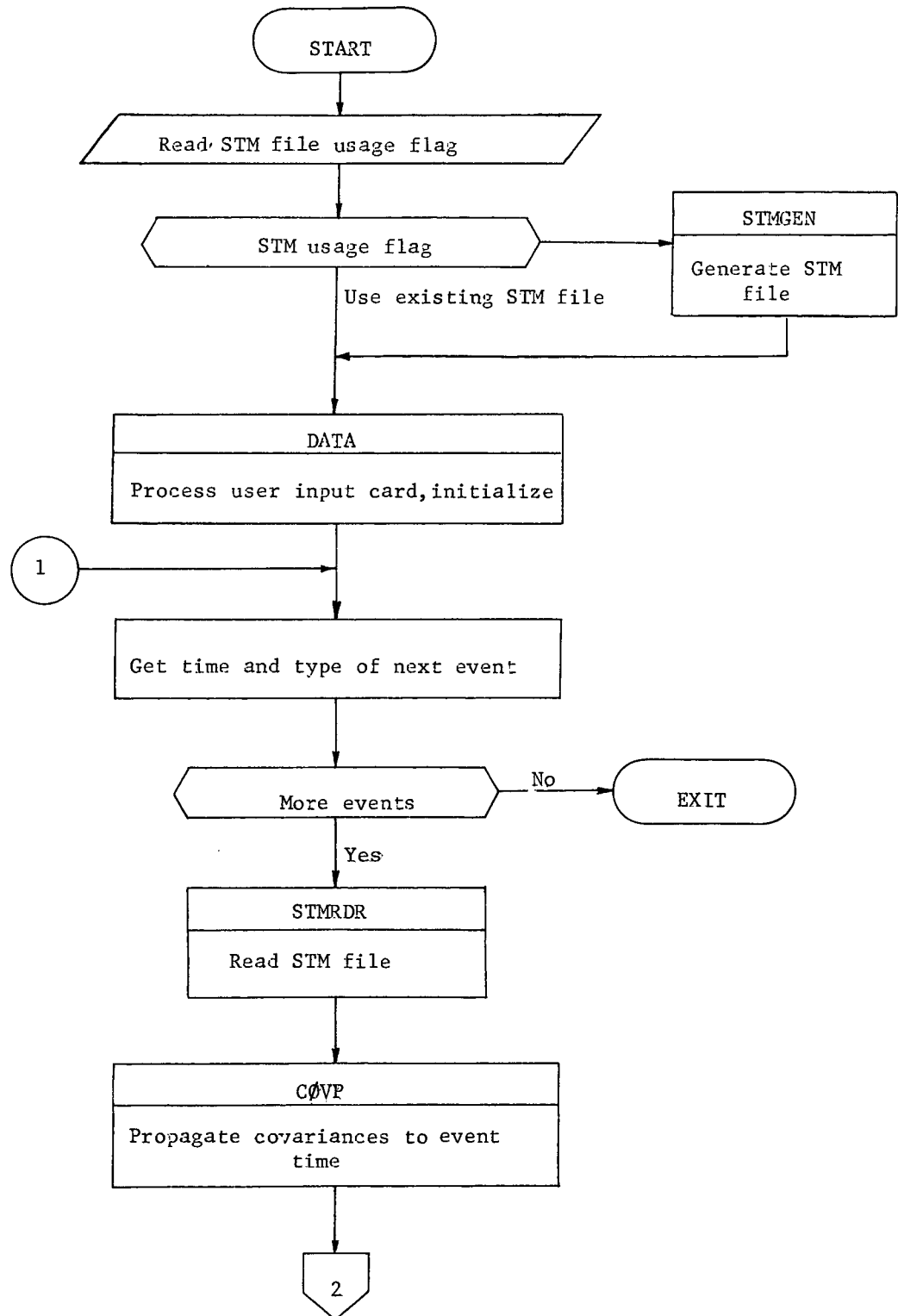


Figure 4. TEAM Macrologic

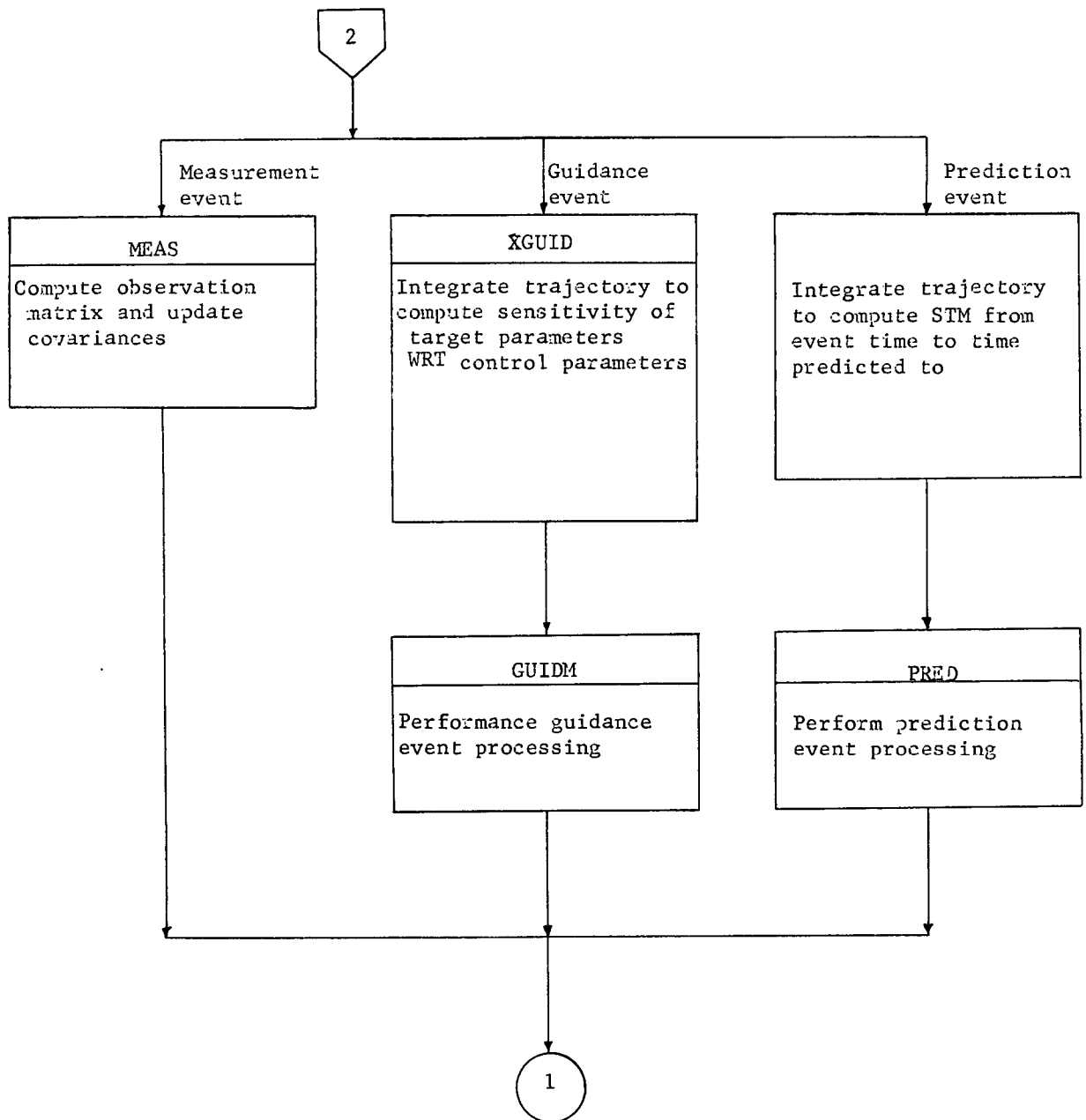


Figure 4. TEAM Macrologic (Continued)

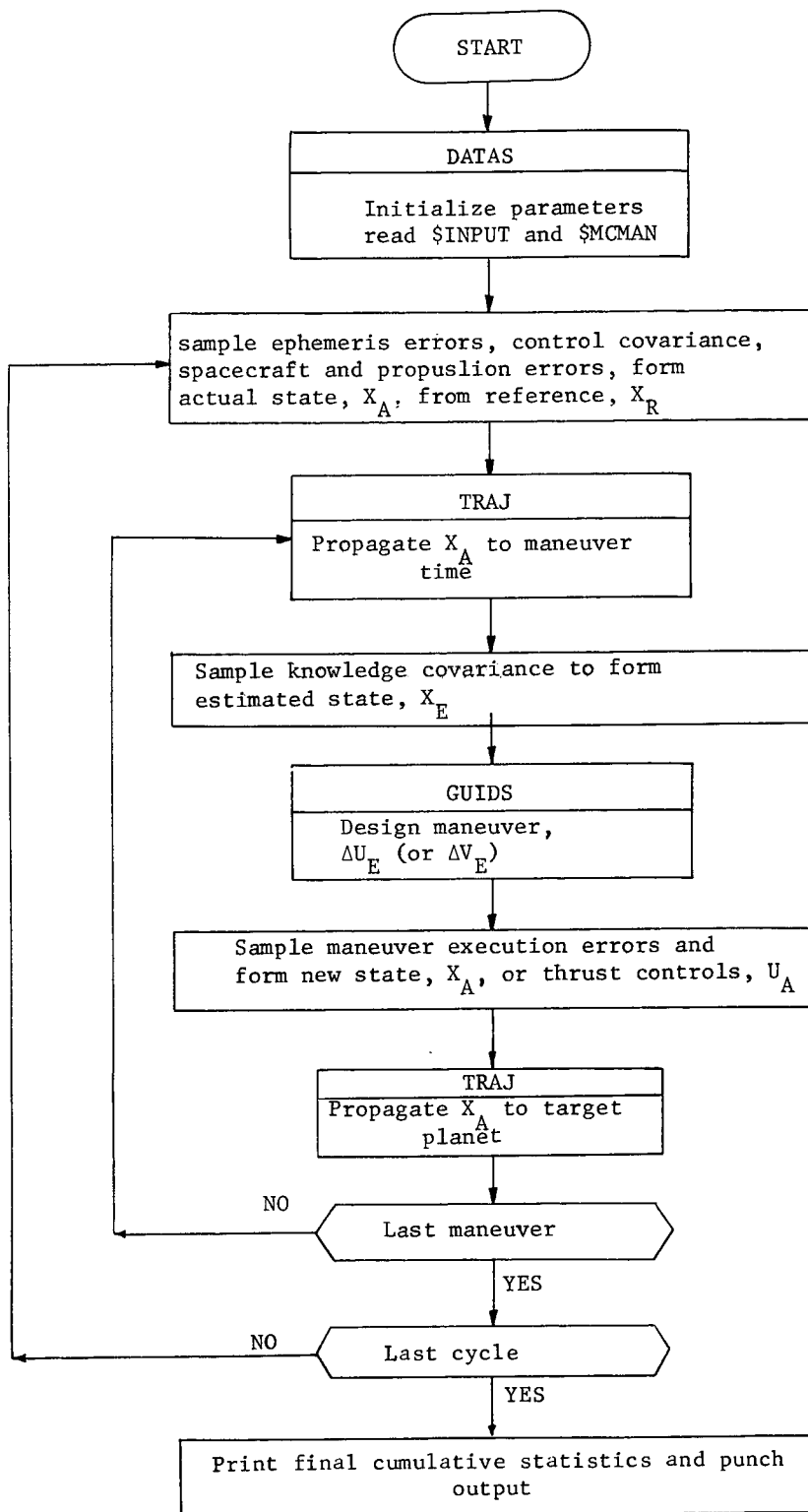


Figure 5. TSIM Macrologic

requirements. Errors are simulated by sampling error source distributions. The actual trajectory is propagated to a guidance or maneuver time where guidance corrections are designed using the estimated spacecraft state. The design maneuver is applied to the actual trajectory after execution errors have been added, and the actual trajectory is propagated to the next maneuver. When all maneuvers have been completed, the actual state is propagated to the target body and actual target conditions are computed. A Monte Carlo analysis is built from repeated passes through this basic cycle and statistical information is computed and printed.

The module consists of a single primary overlay and is called only once for the entire simulation. The functional flow is shown in Figure 5.

4.2 Subroutine Hierarchy

As mentioned previously, FRACAS consists of three independent primary overlays or modes. The subroutine hierarchy for TOM, TEAM, and TSIM are shown in Figures 6,7 and 8, respectively. Multiple calls to subroutines are not shown but may be found in the detailed subroutine descriptions (Section 5). Figure 9 illustrates the trajectory propagation hierarchy which is used in all three modes. Brief descriptions of these subroutines are given below along with references to detailed logic flow to be found in later sections.

SUBROUTINE	PURPOSE	DETAILED DESCRIPTION (SECTION)
BPLANE	compute B-plane parameters	5.1.2.2
BUCKET	sorts elements of a vector	5.2.1
COPY	controls propagation of covariances	5.3.1
CSAMP	determines matrix eigenvectors/values and/or samples covariance	5.4.1
DATA	processes error analysis input data	5.3.2
DATAS	processes simulation input data	5.4.2

SUBROUTINE	PURPOSE	DETAILED DESCRIPTION (SECTION)
DELU	computes control change	5.2.2
DERY	computes covariance derivatives	5.1.2.3
DETECT	detect changes in control	5.1.2.4
DYNO	compute dynamic noise covariance matrix	5.3.3
ENCON	compute or rectify reference conic	5.1.2.5
EP	compute thrust parameters	5.1.2.6
EPHEM	computes inertial state of a natural body	5.1.2.7
FECS	computes performance and target error indices and sensitivities	5.2.3
FILTER	updates knowledge covariance by filtering equations	5.3.4
FUNCT	selects trail steps for trajectory generation	5.2.4
GENMIN	controls curve fitting for scale factor computation	5.2.5
GPRINT	prints true estimation error statistics	5.3.5
GRAVFO	computes gravity gradients and acceleration for primary and perturbing bodies	5.1.2.8
GRID	generates grid of target errors in control space	5.2.6
GUIDM	performs guidance events	5.3.6
GUIDS	designs trajectory correction maneuver	5.4.3
INTEG	performs 4th order Nystrom integration of R, V, and Φ	5.1.2.9
MENØ	computes measurement noise covariance matrix	5.3.9
NØISE	computes acceleration due to time-varying noise	5.4.4

SUBROUTINE	PURPOSE	DETAILED DESCRIPTION (SECTION)
PATH	controls reference trajectory generation; and state transition matrix computation as needed	5.3.10
PINV	computes pseudo inverse of a matrix	
PPOWER	computes power output from low-thrust engine	5.1.2.10
PRED	performs prediction events	5.3.11
PRINT	prints estimated error statistics	5.3.12
PROP	propagates covariances	5.3.13
PTRAN	computes STM	5.3.14
RNUM	generates a Gaussian random number	5.4.5
SCHED	determines time and type of next sequential event	5.3.15
SCOMP	computes sensitivity matrix of target WRT thrust controls	5.4.6
SETEVN	performs computation common to most events	5.3.16
SETUP	stores real-world or assumed constants into working arrays	5.4.7
SIZE	calculates magnitude of control change	5.2.8
STAPRL	computes station location position and velocity partials	5.3.17
STAT	computes cumulative statistics (mean and covariance)	5.4.8
STMGEN	generates STM file	5.3.18
STMHDR	reads STM file	5.3.19
TEAM	controls executive logic flow for error analysis mode	5.3.21

SUBROUTINE	PURPOSE	DETAILED DESCRIPTION (SECTION)
TEST	tests for convergence	5.2.11
TØM	controls I/Ø and initiates targeting/ optimization mode	5.2.12
TRAJ	controls Encke integration	5.1.2.1
TRAKM	computes observation matrices	5.3.22
TSIM	controls logic flow for simulation mode	5.4.10
UPHILL	contols logic flow for targeting/optimization	5.2.13
USRGAN	computes filter gain matrix with user supplied algorithm	5.3.23
WEIGHT	computes weighting matrix	5.2.14
WLSGAN	computes filter gain matrix according to sequential weighted least squares algorithm	5.3.24
XGUID	controls execution sequence for guidance events	5.3.25

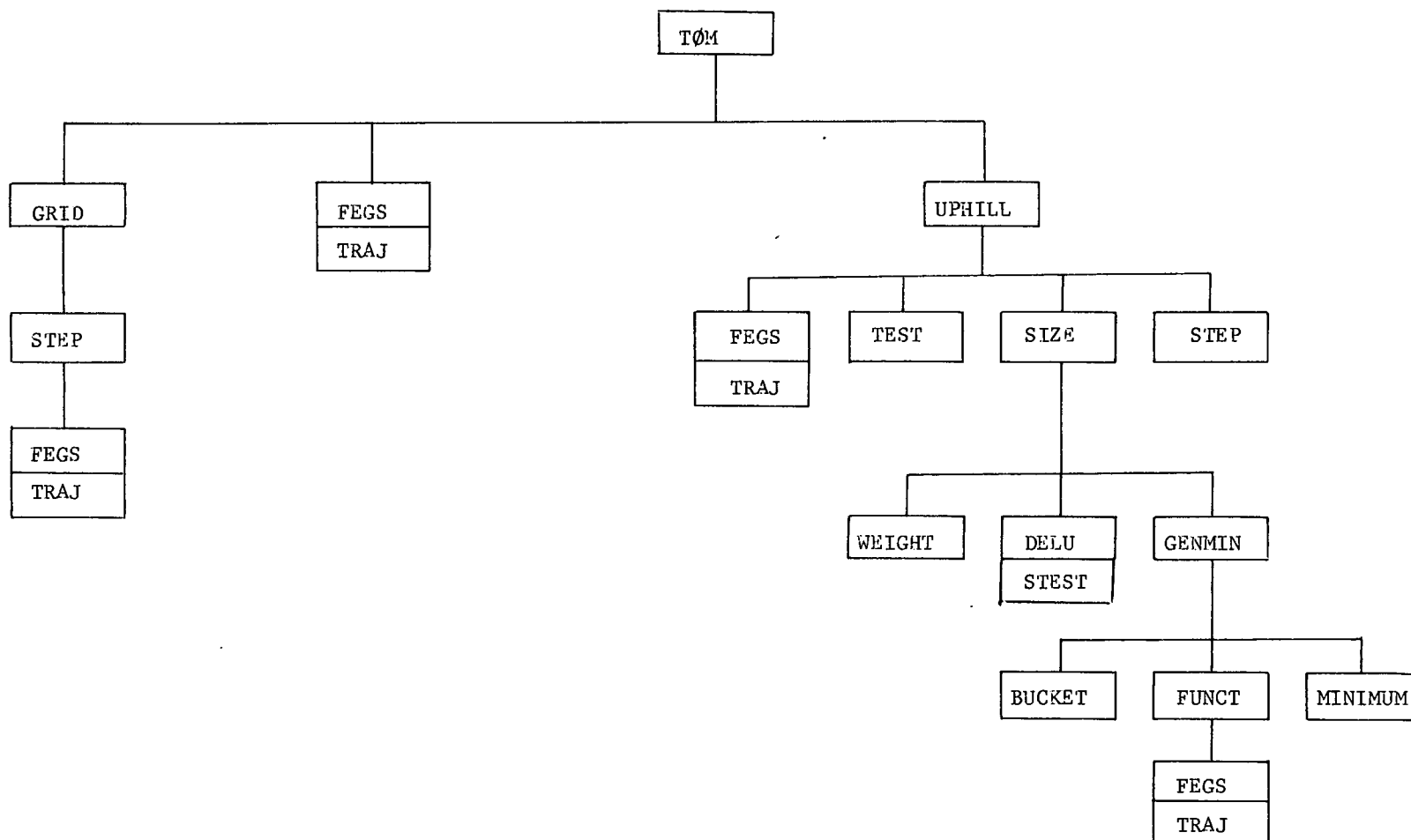


FIGURE 6. Subroutine Hierarchy for Targeting and Optimization Mode

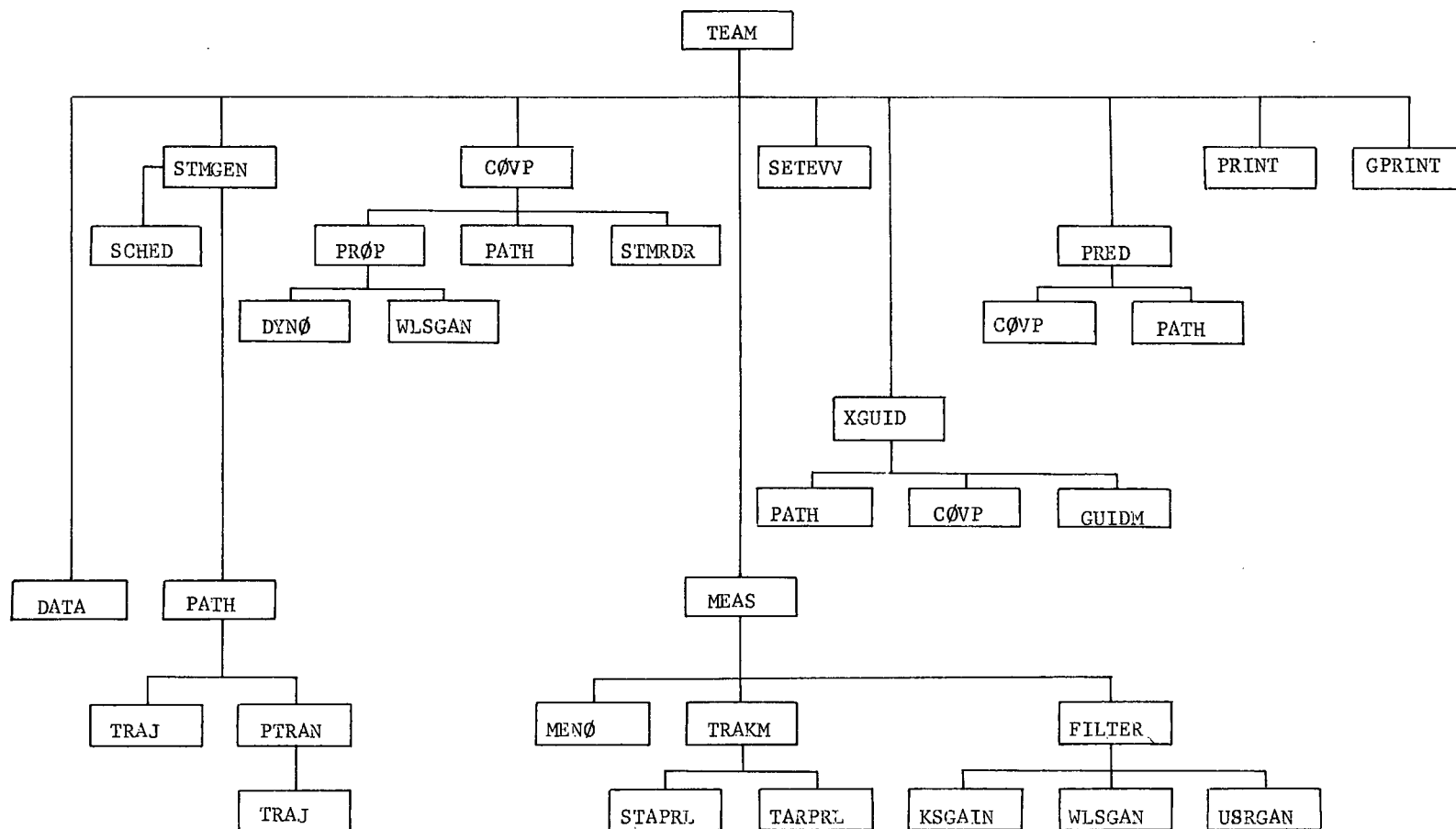


FIGURE 7. Subroutine Hierarchy for Error Analysis Mode

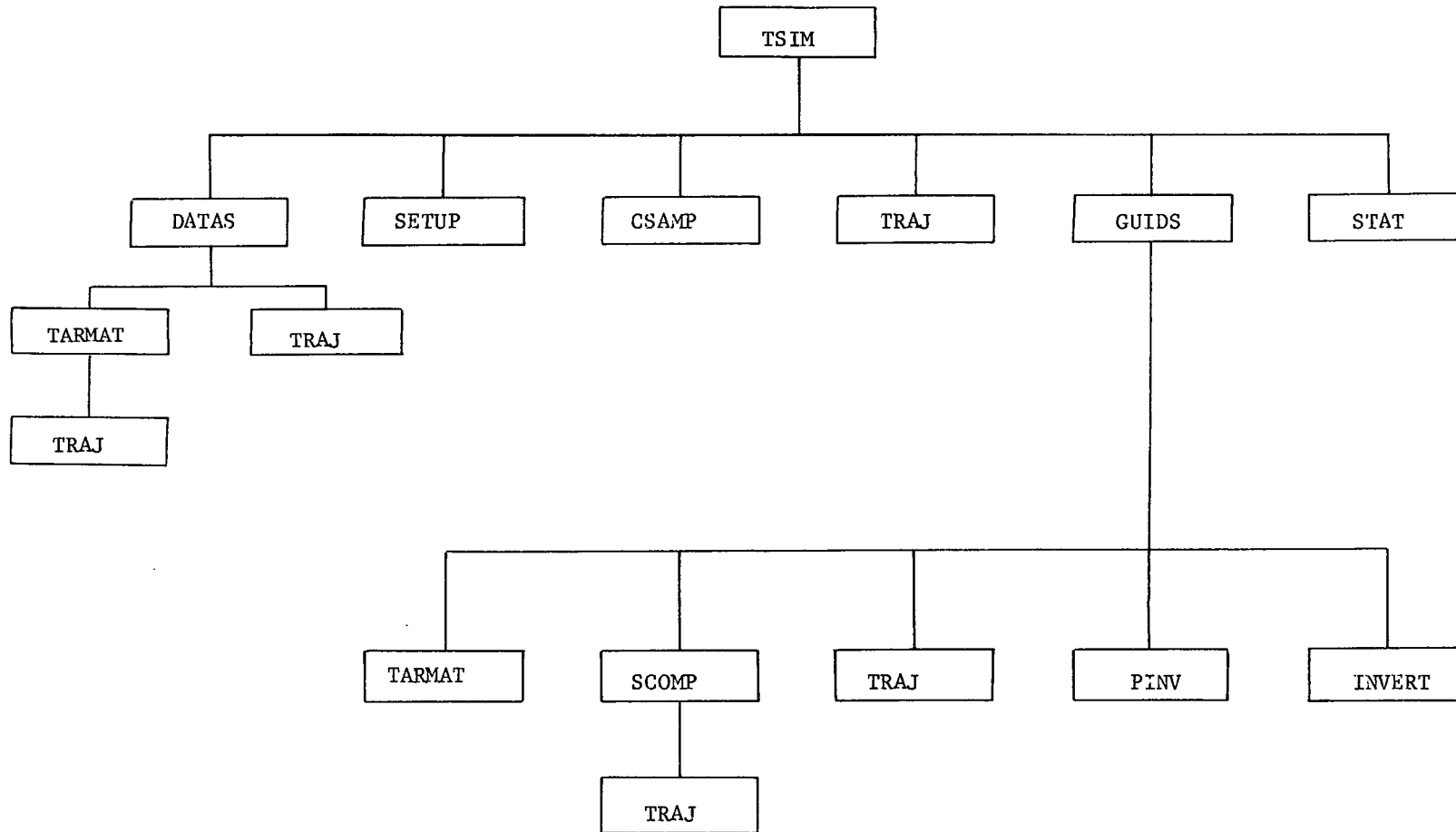


FIGURE 8. Subroutine hierarchy for Trajectory Simulation Mode

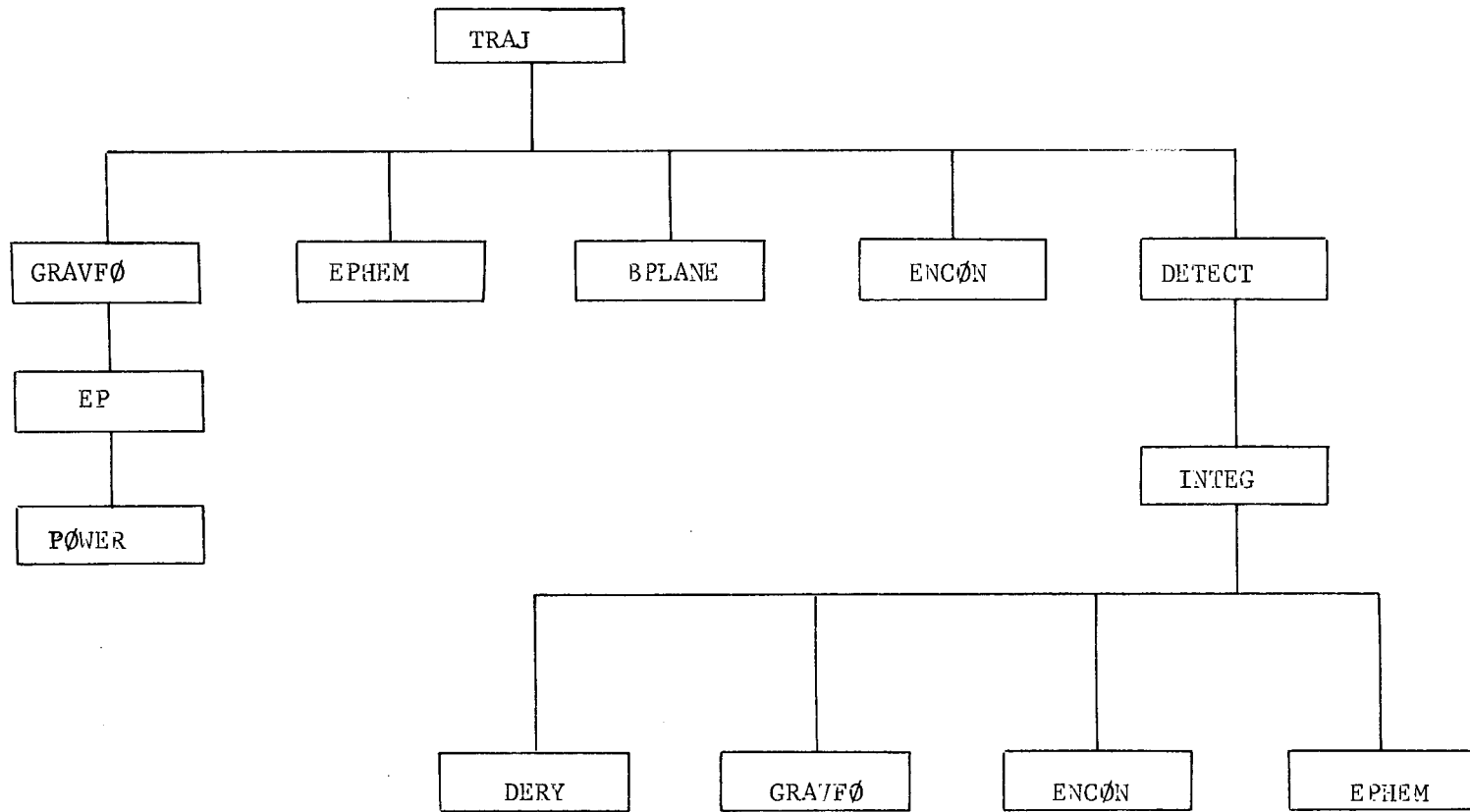


FIGURE 9. Subroutine Hierarchy for Trajectory Propagation

5 SUBROUTINE DESCRIPTIONS

5.1 Subroutines Used in More than One Mode

5.1.1 Utility Routines

Many small routines are used by several of the FRACAS modules.

These are described briefly below:

- o ADD - add two matrices
- o ADPR - additive matrix product, $[A] = [A] + [B] [C]$
- o ADPRT - additive matrix product $[A] = [A] + [B] [C]^T$
- o ADXYXT - additive matrix product $[A] = [A] + [B] [C] [B]^T$
- o CØPY - copy one matrix into another
- o CØPYT - copy the transpose of one matrix into another
- o CØRREL - compute standard deviations and correlations of
 a covariance
- o DDØTB - compute the inner product of two vectors
- o DUMAG - compute the magnitude of a vector
- o DXB - compute the cross product of two vectors
- o GHA - compute the Greenwich hour angle
- o INVERT - invert a matrix
- o JACØBI - compute the eigenvectors (eigenvalues of a matrix)
- o MATØUT - print a matrix
- o MULT - matrix product $[A] = [B] [C]$
- o MULTT - matrix product $[A] = [B] [C]^T$
- o PINV - pseudo inverse of a matrix
- o SDCØV - converts standard deviations and correlations
 to covariance
- o SUB - subtract one matrix from another $[A] = [B] - [C]$
- o SUBT - subtract one matrix from another $[A] = [B] - [C]^T$
- o SYMTRZ - symmetrize a matrix

- o TADPR - additive matrix product $[A] = [A] + [B]^T [C]$
- o TADPRT - additive matrix product $[A] = [A] + [B]^T [C]^T$
- o TIM - convert time in seconds to days, hours, minutes,
and seconds
- o TMULT - matrix product $[A] = [B]^T [C]$
- o TMULTT - matrix product $[A] = [B]^T [C]^T$
- o UNITV - unitize a vector
- o XTYX - matrix triple product $[A] = [B]^T [C] [B]$
- o XYXT - matrix triple product $[A] = [B] [C] [B]^T$
- o ZMAT - zero out a matrix

5.1.1.1 Subroutine: CONIC

Purpose: To convert cartesian coordinates to conic elements

Input:

- o position, \underline{r}
- o velocity, \underline{v}
- o gravitational constant of primary body, μ

Output:

- o semi-major axis, a
- o eccentricity, e
- o inclination, i
- o longitude of ascending node, Ω
- o argument of periapsis, ω
- o mean anomaly, M

Remarks: let $\underline{h} = \underline{r} \times \underline{v}$

$$\underline{w} = \hat{\underline{h}}$$

$$d = \underline{r} \cdot \underline{v}$$

$$\underline{c} = \frac{1}{\mu} (\underline{v} \times \underline{h}) - \underline{r}$$

$$\underline{p} = \underline{c}$$

$$s = |\underline{h}|/\mu$$

then $i = \cos^{-1}(\underline{w}_z)$

$$\Omega = \tan^{-1} \frac{\underline{w}_x}{-\underline{w}_y}$$

$$\underline{q} = \underline{w} \times \underline{p}$$

$$\omega = \tan^{-1} (\underline{p}_z / \underline{q}_z)$$

$$\sin(\theta) = (|\underline{h}|d)/|\underline{r}|$$

$$\cos(\theta) = (|\underline{h}|^2 - \mu)/|\underline{r}|$$

$$\theta = \tan^{-1}(\sin(\theta)/\cos(\theta))$$

$$\beta = \cos^{-1}(d/(|\underline{r}||\underline{v}|))$$

$$\cos(E) = 1 - \frac{|r|}{a}$$

$$\sin(E) = d / \mu |a|$$

for elliptical case ($a > 0$)

$$E = \tan^{-1}(\sin(E)/\cos(E))$$

$$M = E - \sin(E)$$

for hyperbolic case ($a < 0$)

$$\sinh(f) = \sin(E) / |c|$$

$$\cosh(f) = \cos(E) / |c|$$

$$E = \ln(\sinh(f) + \cosh(f))$$

$$M = \sin(E) - E$$

5.1.1.2 Subroutine CØNVRT

Purpose: To convert spherical coordinates to cartesian coordinates.

Input: o spherical coordinates of position (r, Ø, Ø)
 o spherical coordinates of velocity (v, γ, σ)

Output: cartesian position vector \vec{R}
 cartesian position vector \vec{V}

Remarks:

$$R_x = r \cos\emptyset \cos\theta$$

$$R_y = r \cos\emptyset \sin\theta$$

$$R_z = r \sin\emptyset$$

$$B_x = v \sin\gamma$$

$$B_y = v \cos\gamma \sin\sigma$$

$$B_z = v \cos\gamma \cos\sigma$$

$$V_x = B_x \cos\emptyset \cos\theta - B_y \sin\theta - B_z \sin\emptyset \cos\theta$$

$$V_y = B_x \cos\emptyset \sin\theta + B_y \cos\theta - B_z \sin\emptyset \sin\theta$$

$$V_z = B_x \sin\emptyset + B_z \cos\emptyset$$

5.1.1.3 Subroutine EULMX

Purpose: To compute a rotational transformation matrix from the Euler angles.

Input: o Euler angles (α, β, γ)
 o axes of rotation, a_i , $i = 1, 3$

Output: o transformation matrix $[P]$

Remarks:

$$[P] = [H][G][F]$$

where $[F] = f(\alpha, a_1)$

$$[G] = f(\beta, a_2)$$

$$[H] = f(\gamma, a_3)$$

$f(\psi, a)$ is defined as

$$\text{for } a = 1, f(\psi, a) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

$$\text{for } a = 2, f(\psi, a) = \begin{bmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{bmatrix}$$

$$\text{for } a = 3, f(\psi, a) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.1.1.4 Subroutine PECEQ

Purpose: To compute the transformation matrix from
planetoecentric ecliptic to planetocentric
equatorial coordinates,

Input:

- o planet number
- o Julian date
- o planets conic elements, $(a, e, i, \Omega, \omega, M)$
- o planets right ascension α and declination δ of
the pole vector
- o obliquity of the ecliptic, ϵ

Output: transformation matrix $[A]$

Remarks:

Let \hat{P} be the planetary pole vector,

$$\hat{P} = \begin{bmatrix} \cos \alpha \cos \delta \\ \cos \epsilon \sin \alpha \cos \delta + \sin \epsilon \sin \delta \\ -\sin \epsilon \sin \alpha \cos \delta + \cos \epsilon \sin \delta \end{bmatrix}$$

and $\hat{N} = \begin{bmatrix} \sin i \sin \Omega \\ -\sin i \cos \Omega \\ \cos i \end{bmatrix}$

then $[A] = [\hat{X} \ \hat{Y} \ \hat{Z}]^T$

where $\hat{Z} = \hat{P}$

$$\hat{X} = \hat{P} \times \hat{N} / |\hat{P} \times \hat{N}|$$

$$\hat{Y} = \hat{Z} \times \hat{X}$$

5.1.2 Trajectory Routines

5.1.2.1 Subroutine TRAJ

Purpose: To control the integration of the trajectory (and certain other parameters) between two time points

Input:

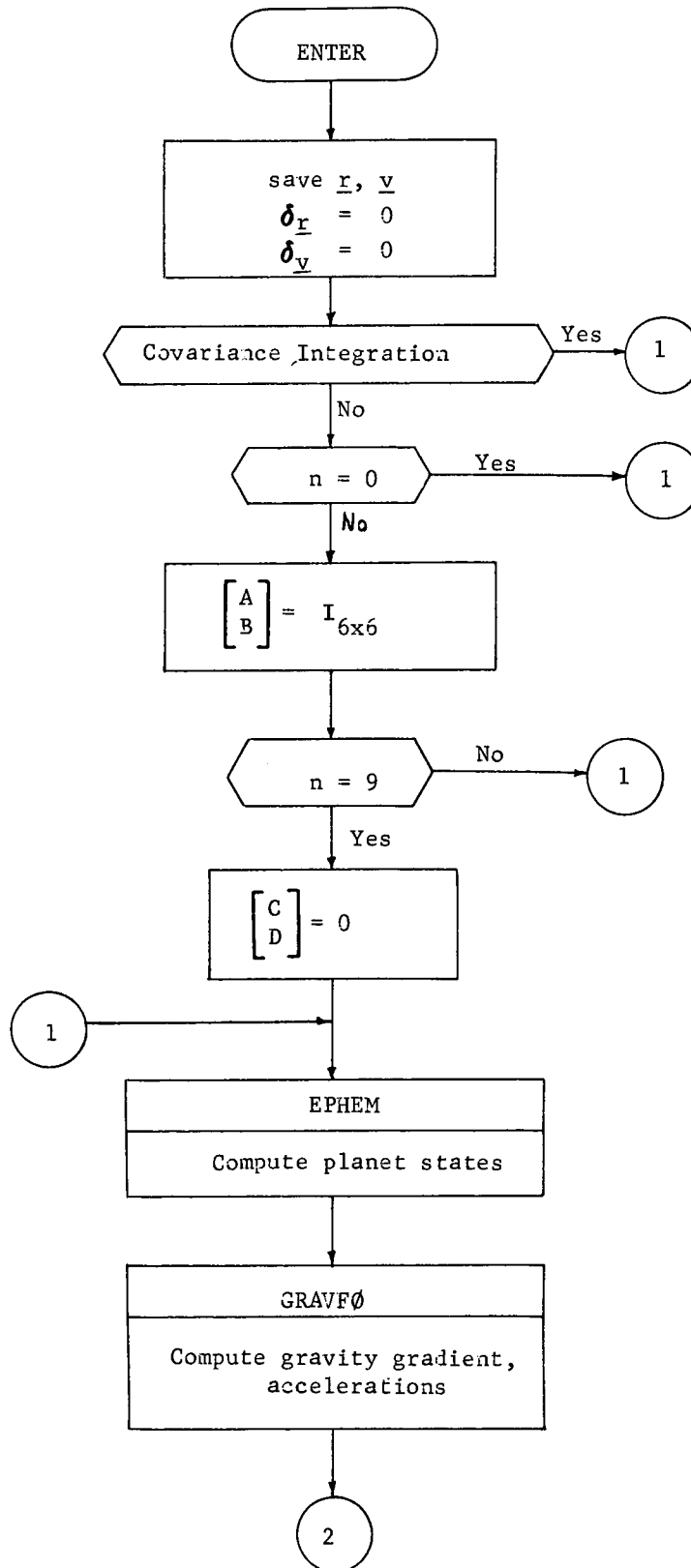
- o thrust controls
- o true state $\underline{r}, \underline{v}$
- o covariance integration flag
- o primary body
- o target planet
- o start time, t_k
- o stop time, t_{k+1}
- o dimension of state transition matrix, n

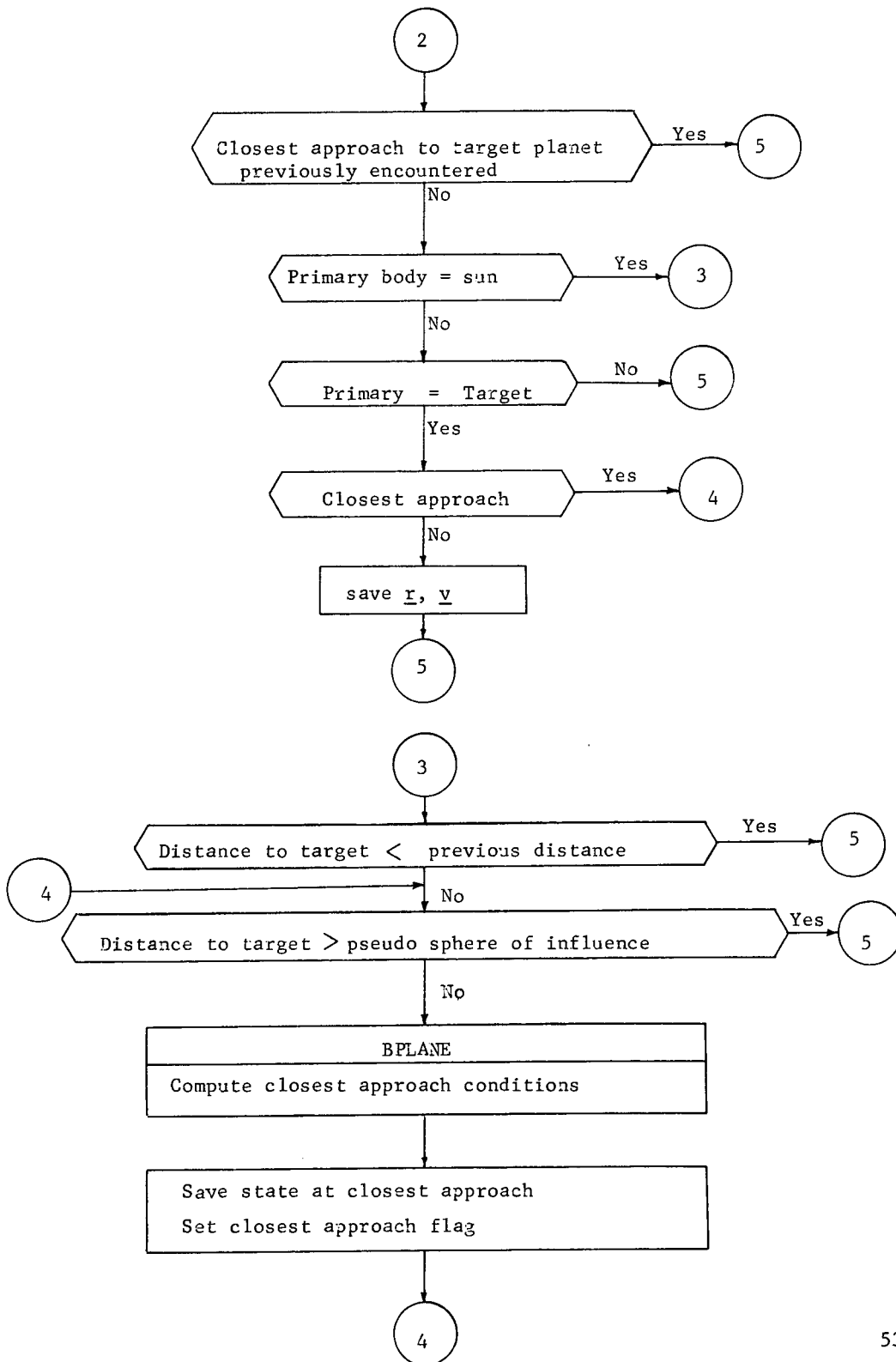
Output:

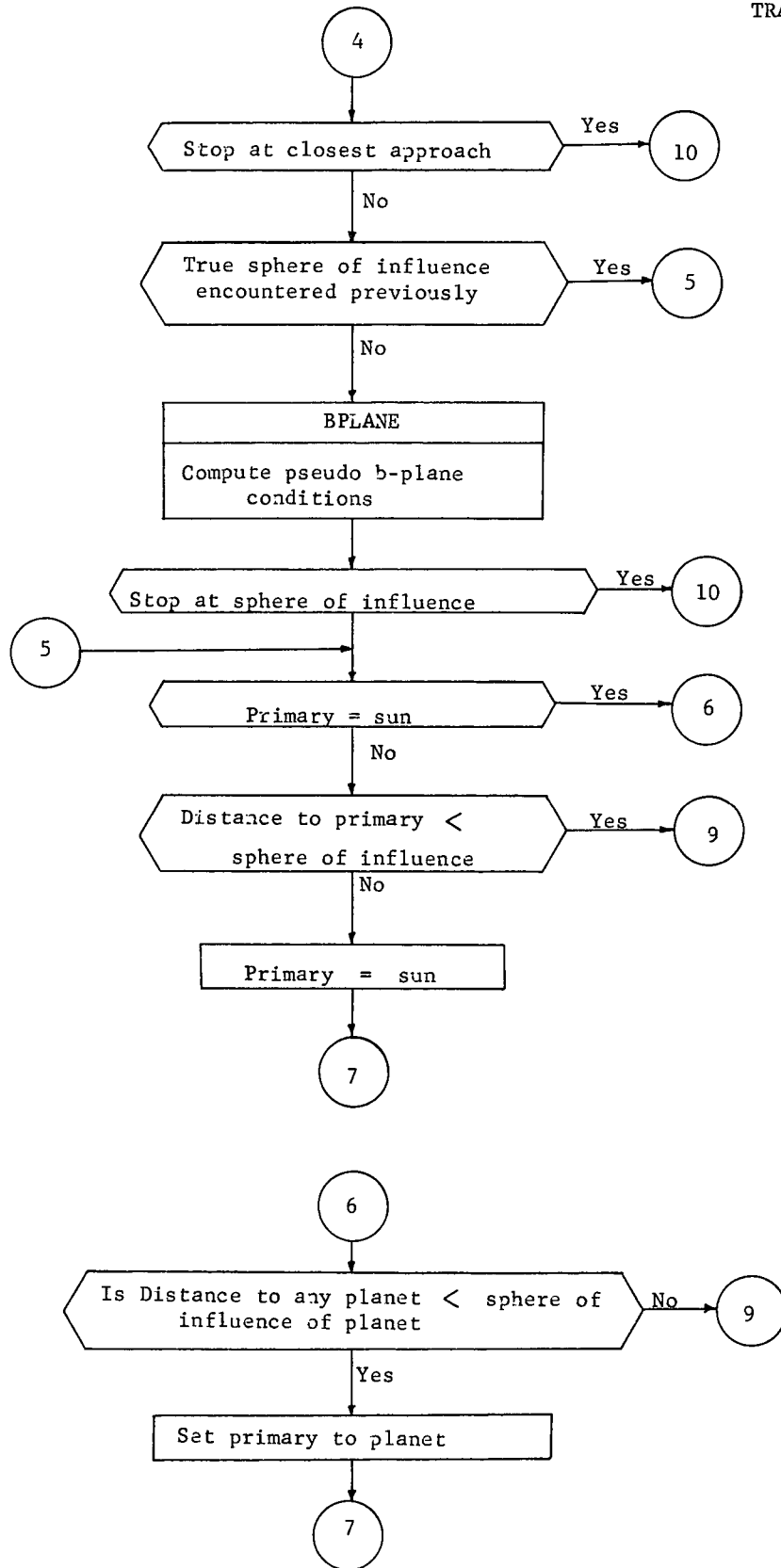
- o true state at t_{k+1}
- o integrated state transition matrix or augmented state covariance matrix

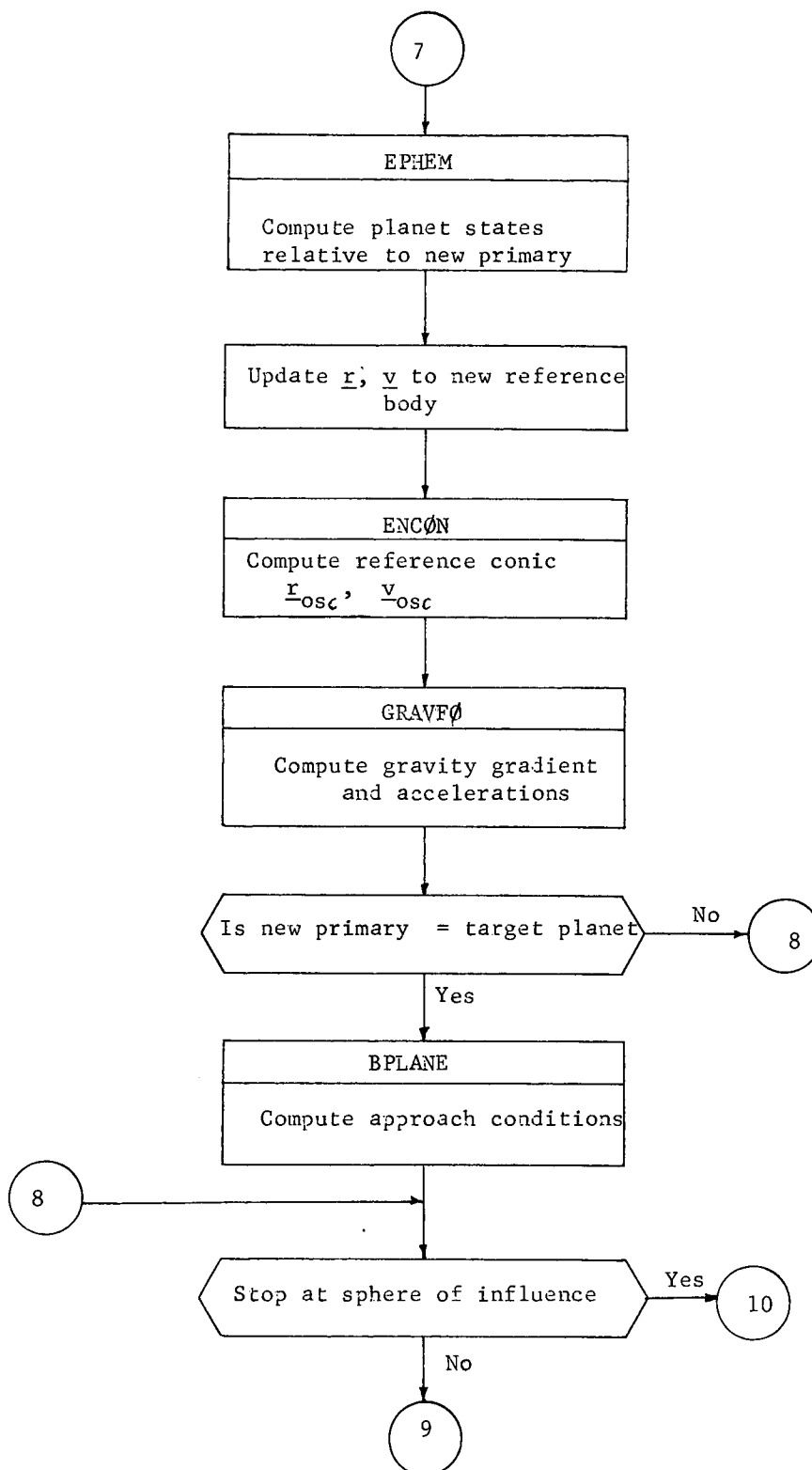
Remarks:

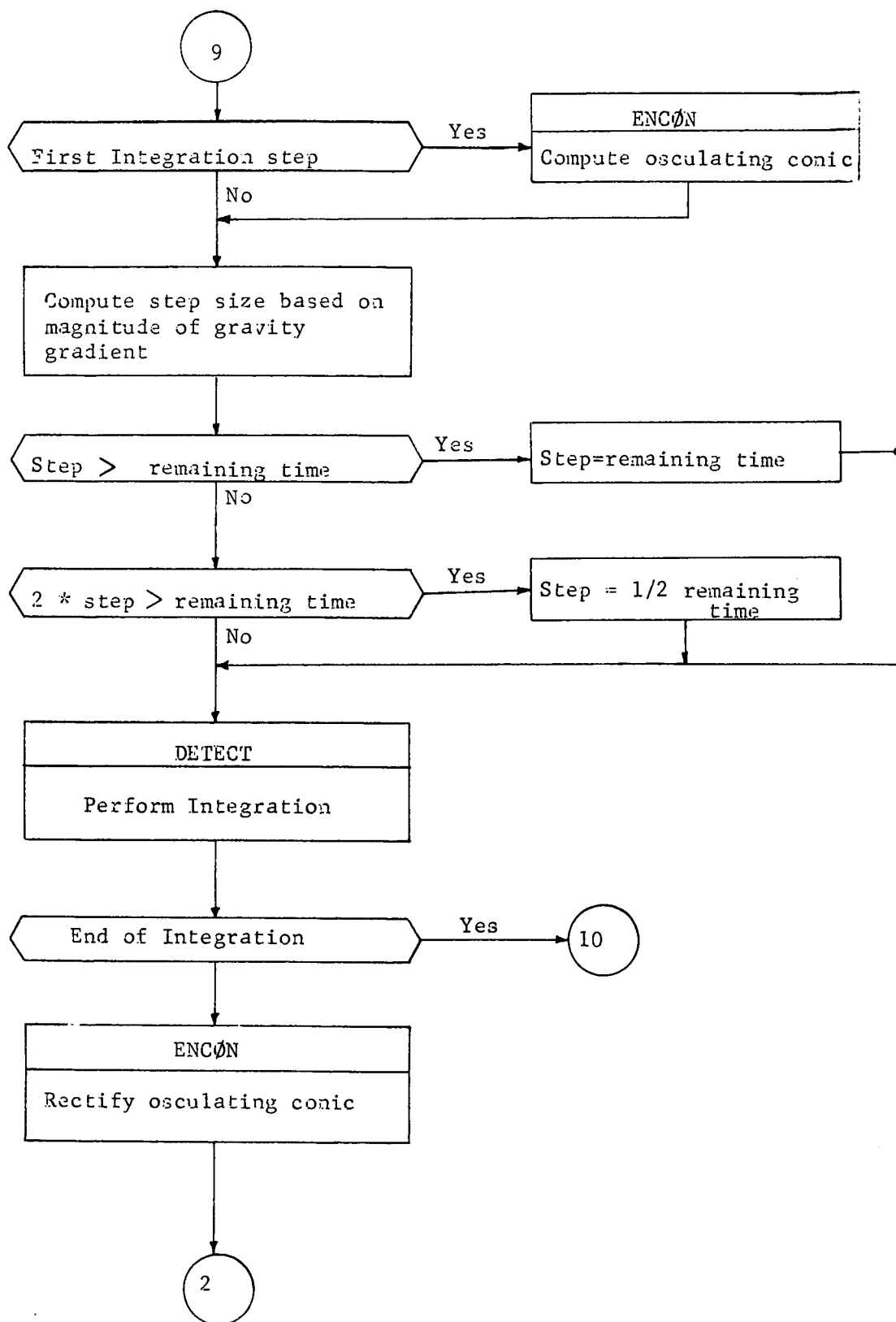
TRAJ is the logic control routine for the integrator. The Encke perturbed conic method (Ref. Battin ch. 6) is used with a Nystrom fourth-order two-step numerical integration technique. TRAJ can optionally integrate the covariances directly or compute state transition matrices for either the basic state or the state augmented by thrust parameters.

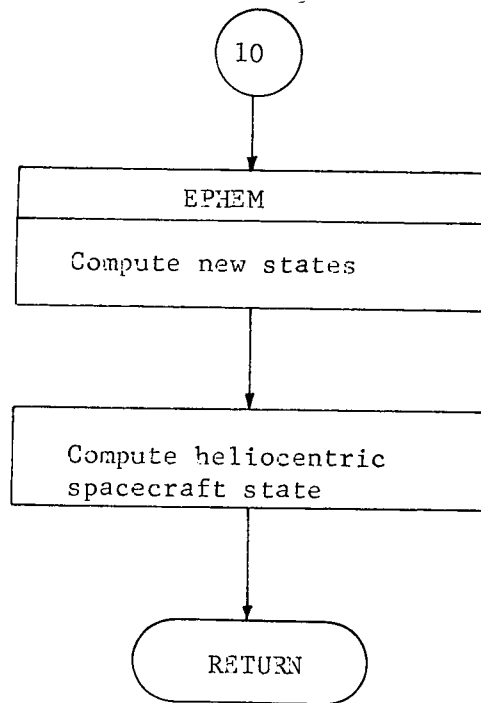












5.1.2.3 Subroutine DERY

Purpose: To compute the derivative of the augmented state covariance matrix.

Input:

- o augmented state covariance matrix $[P]$
- o thrust transformation matrix $[T]$
- o gravity gradient $[G]$
- o process noise correlation times, τ_i , $i=1,6$

Output: o augmented state covariance derivative matrix $[\dot{P}]$

Remarks:

Let $P = E \begin{bmatrix} X & X^T \end{bmatrix}$
 where $X = \begin{bmatrix} \underline{p} \\ \underline{v} \\ \underline{n} \\ \underline{u} \end{bmatrix}$

\underline{p} = deviations of position components from nominal

\underline{v} = deviations of velocity components from nominal

\underline{n} = deviations of thrust components due to noise

\underline{u} = deviations of thrust components due to bias

then $\dot{P} = [F] [P] + [P] [F]^T$

where P is partitioned

$$P = \begin{bmatrix} P_p & C_{pv}^T & C_{pn}^T & C_{pu}^T \\ C_{pv} & P_v & C_{vn}^T & C_{vu}^T \\ C_{pn} & C_{vn} & P_n & C_{nu}^T \\ C_{pu} & C_{vu} & C_{nu} & P_u \end{bmatrix}$$

and F is partitioned

$$F = \begin{bmatrix} 0 & I & 0 & 0 \\ G & 0 & N & T \\ 0 & 0 & H & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $N = [G \ G]$

$$H = \begin{bmatrix} -1/\tau_1 & 0 \\ 0 & -1/\tau_6 \end{bmatrix}$$

then $[\dot{P}_p] = [C_{pv}] + [C_{pv}]^T$

$$[\dot{C}_{pv}] = [G][P_p] + [N][C_{pn}] + [T][C_{pu}] + [P_v]$$

$$[\dot{C}_{pn}] = [H][C_{pn}] + [C_{vn}]$$

$$[\dot{C}_{pu}] = [C_{vu}]$$

$$[\dot{P}_v] = [G][C_{pv}]^T + [N][C_{vn}] + [T][C_{vu}] + [C_{pv}][G]^T + [C_{vn}]^T[N]^T + [C_{vu}]^T[G]^T$$

$$[\dot{C}_v] = [H][C_v] + [C_p][G]^T + [P_n][N]^T + [C_{nu}]^T[G]^T$$

$$[\dot{C}_{vu}] = [C_{pu}][G]^T + [C_{nu}][N]^T + [P_u][G]^T$$

$$[\dot{P}_n] = 0$$

$$[\dot{C}_{nu}] = [H][C_{nu}]$$

$$[\dot{P}_u] = 0$$

5.1.2.4 Subroutine DETECT

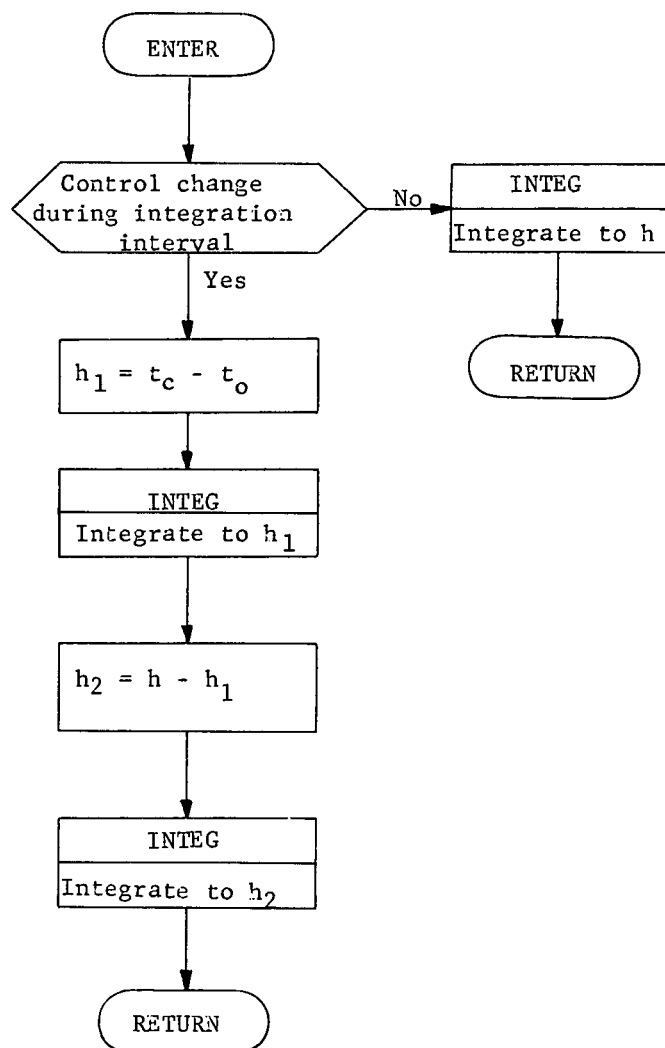
Purpose To detect control changes during an integration step and break up the step at the time of change

Input:

- current time t_o
- proposed step size h
- control times T_c

Output: None

Logic flow:



5.1.2.5 Subroutine ENCØN

Purpose: To propagate reference conic to current time or rectify conic if deviations are too large.

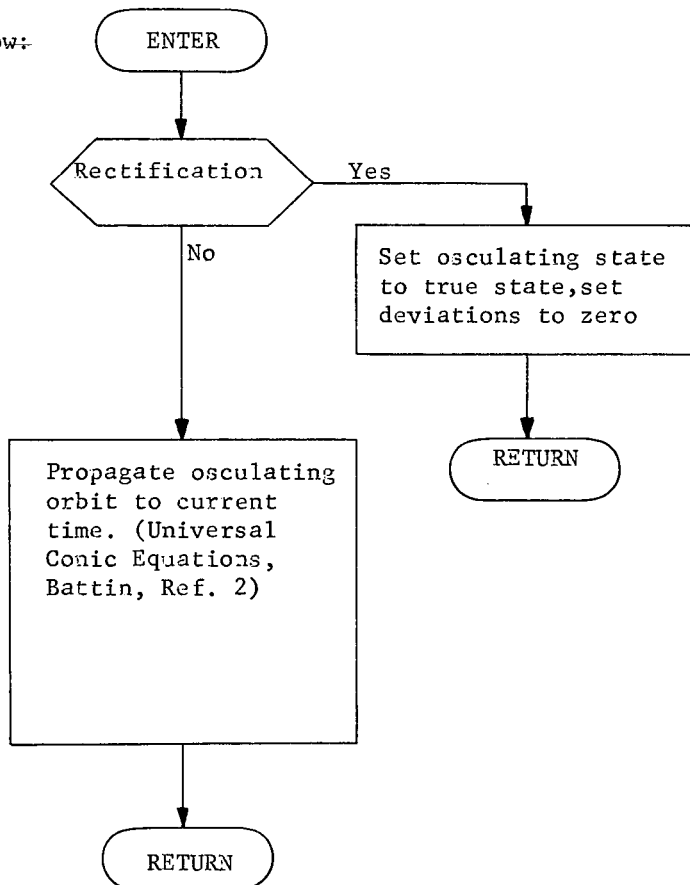
Input:

- true position vector
- true velocity vector
- osculating conic
- previous time

Output:

- updated osculating position vector
- updated osculating velocity vector
- osculating

Logic flow:-



5.1.2.6 Subroutine EP

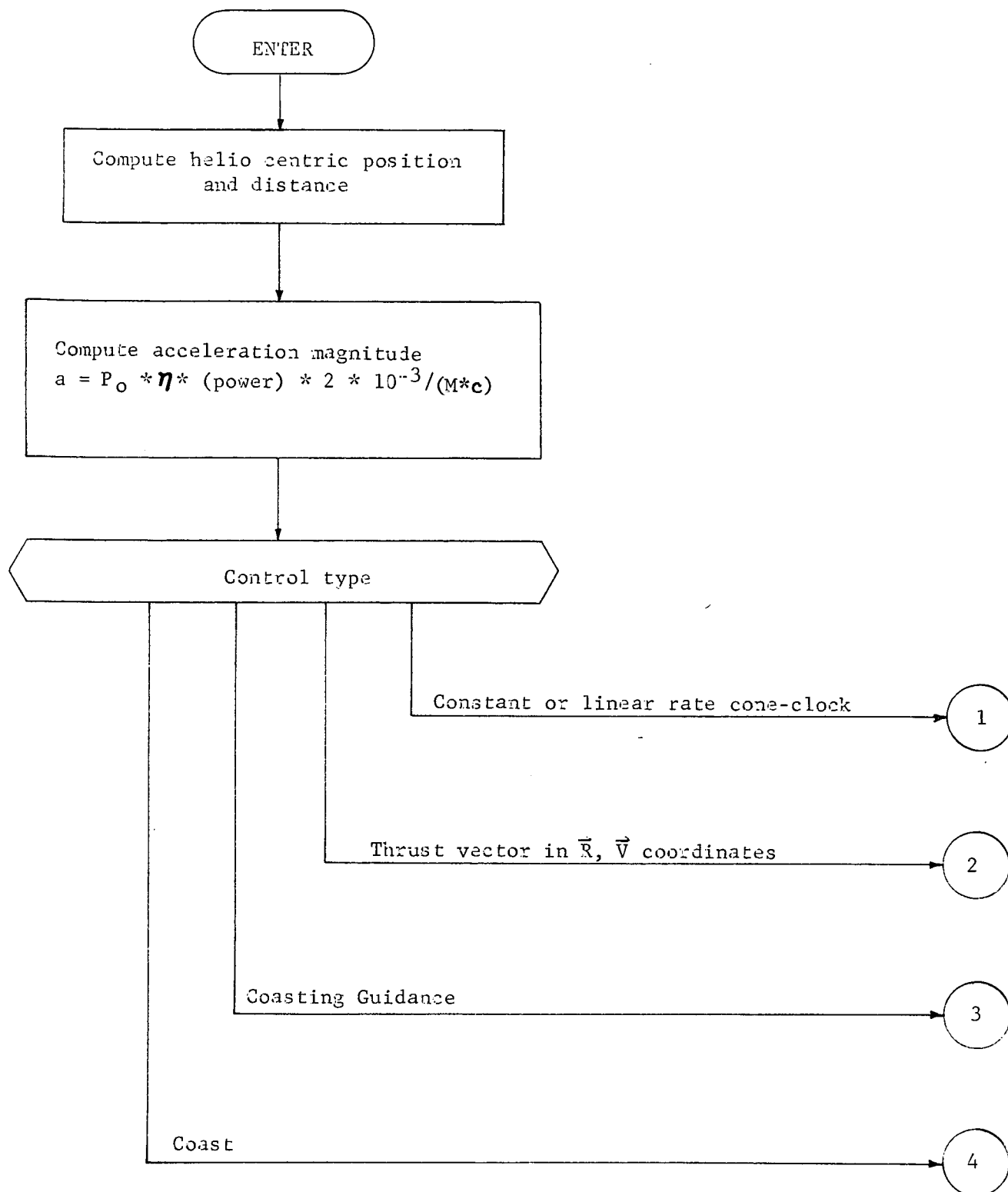
Purpose: To compute magnitude and direction of low thrust.

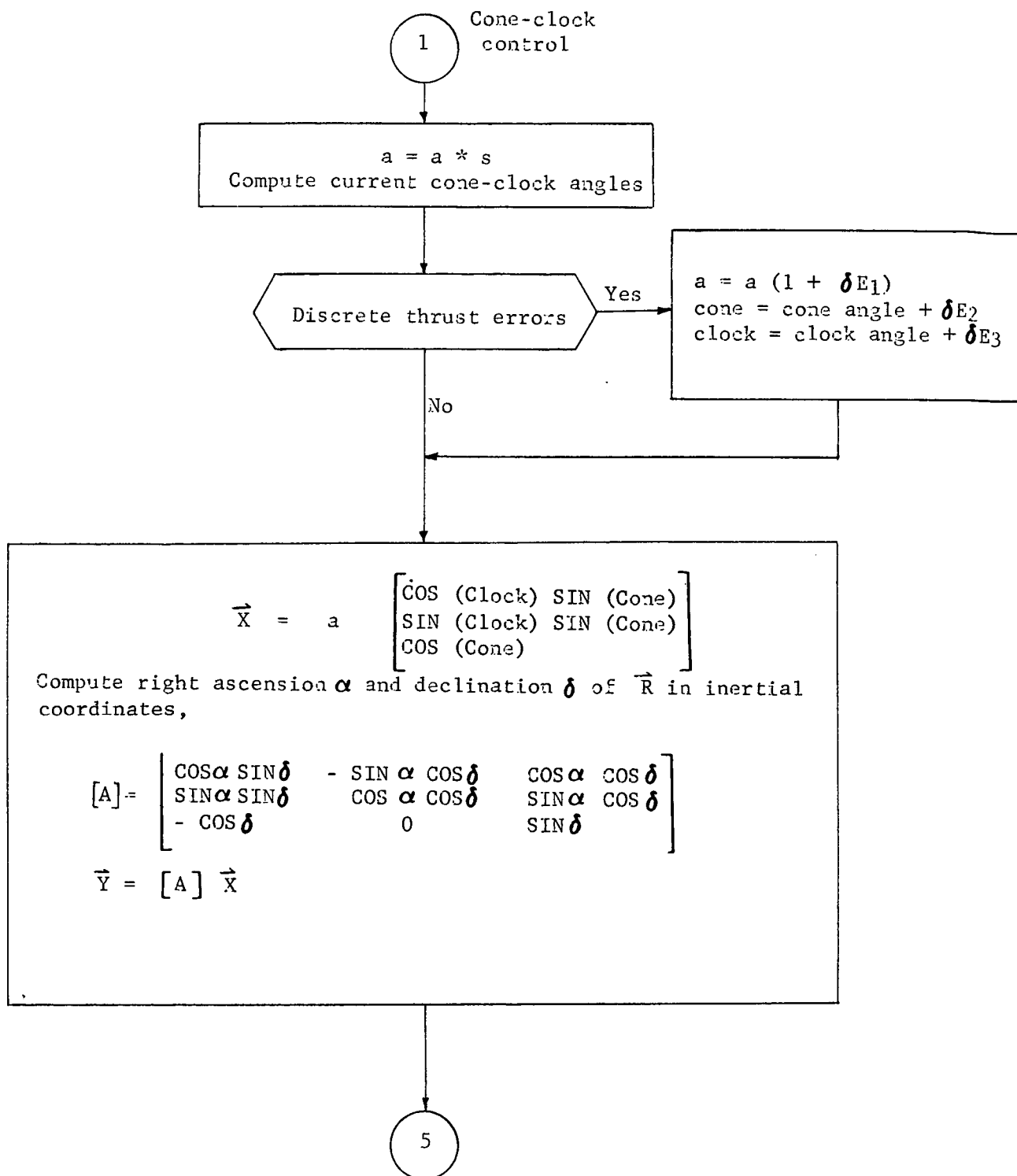
Input:

- o spacecraft mass, M
- o exhaust velocity, c
- o base power, P_o
- o engine efficiency, η
- o spacecraft position, \vec{R}
- o spacecraft velocity, \vec{V}
- o thrust controls: control type, thrust scale factor(s)
- o simulation flag
- o simulation error levels, δE_i , $i=1,3$

Output:

- o thrust vector, \vec{y}
- o thrust vector rotation matrix into inertial coordinates, $[T]$





2

\vec{R}, \vec{V}
control

$$a = a * s$$

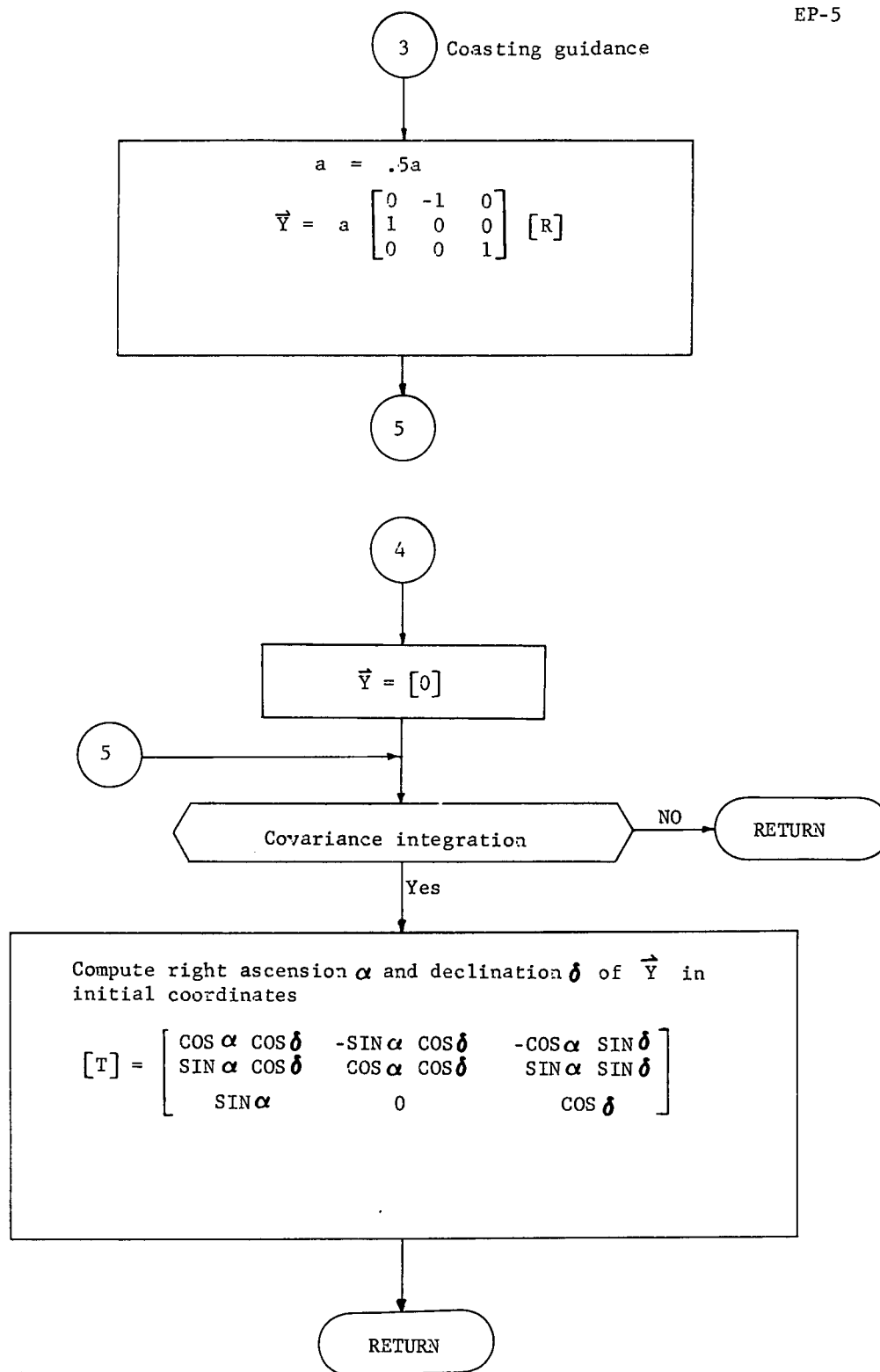
Compute current in-out plane angles

$$\vec{X} = a \begin{bmatrix} \cos(\text{Out}) \cos(\text{in}) \\ \cos(\text{Out}) \sin(\text{in}) \\ \sin(\text{Out}) \end{bmatrix}$$

$$[A] = \begin{bmatrix} \frac{\vec{V} \cdot (\vec{R} \times \vec{V}) \times \vec{V}}{|\vec{V}| \cdot |\vec{R} \times \vec{V}| \times |\vec{V}|} & \frac{\vec{R} \times \vec{V}}{|\vec{R} \times \vec{V}|} \end{bmatrix}$$

$$\vec{Y} = [A] \vec{X}$$

5



5.1.2.7 Subroutine EPHEM

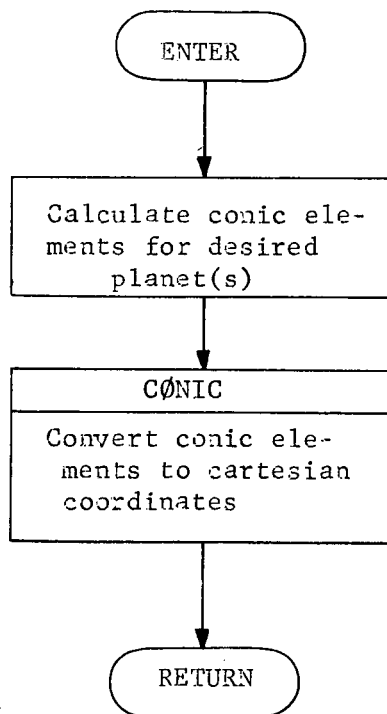
Purpose: To calculate the position and velocity of a planet

Input:

- Julian date
- planet code(s)
- planetary constants

Output: heliocentric state vector of planet

Logic flow:



5.1.2.8 Subroutine GRAVFØ

Purpose: To compute gravity gradient of primary body and perturbing bodies and to compute accelerations caused by low thrust propulsion.

Input:

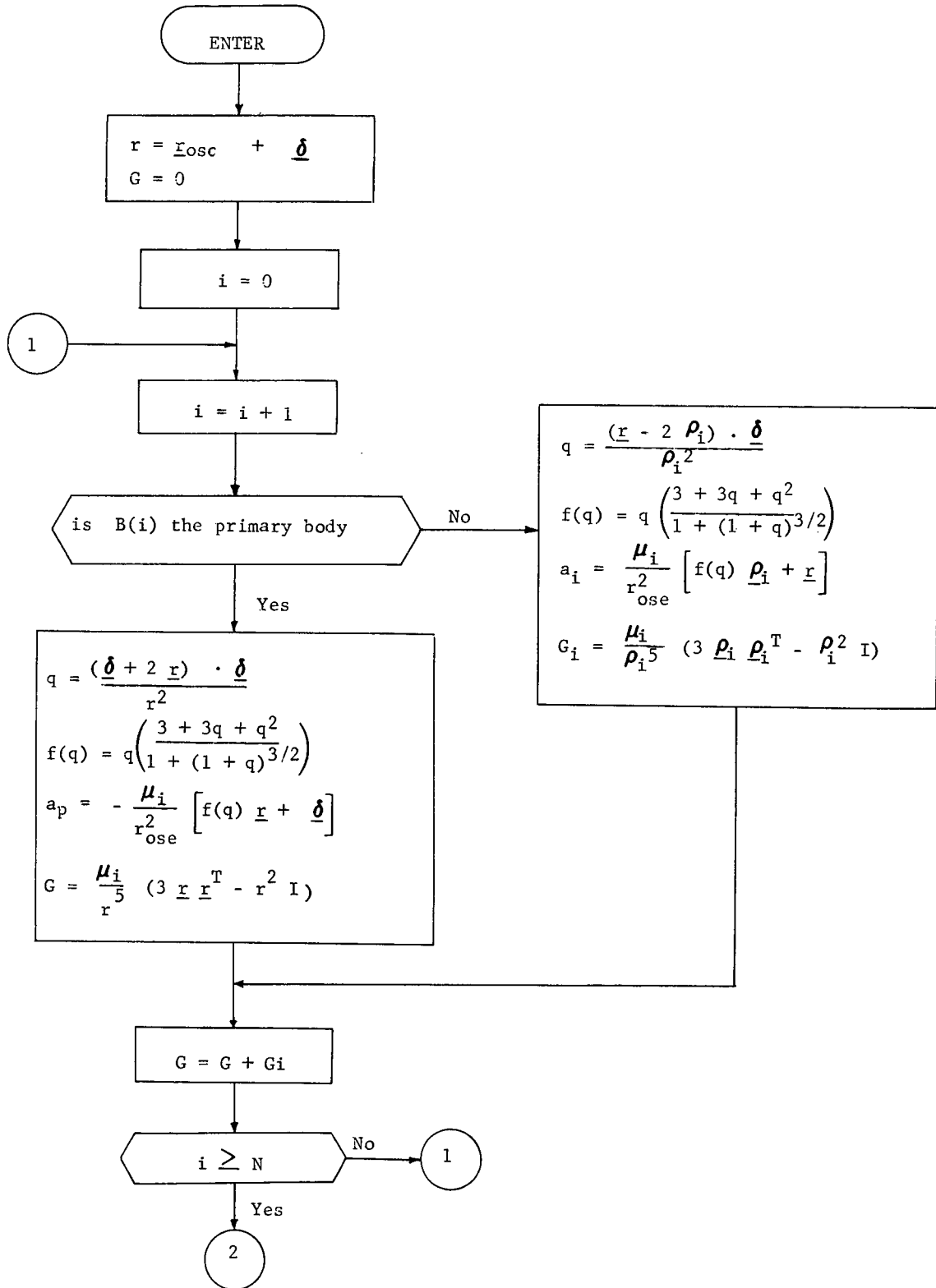
- o osculating spacecraft position vector relative to primary body \underline{r}_{osc}
- o difference between osculating position and true position, $\underline{\delta}$
- o planets or bodies to be considered, $B(i)$, $i=1, \dots, N$
- o positions of perturbing bodies relative to primary, ρ_i

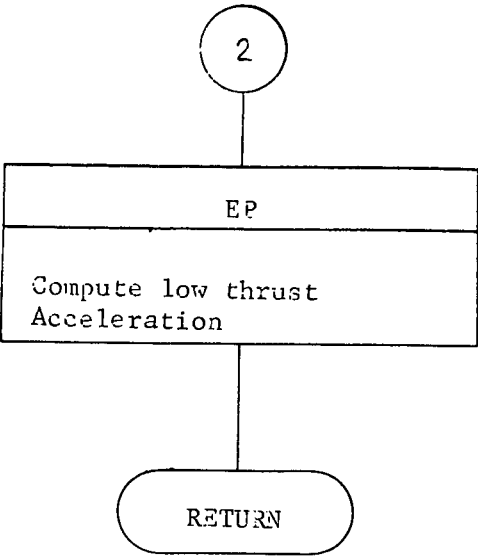
Output:

- o gravity gradient matrix, G
- o acceleration due to perturbing bodies and thrust, a_d

Logic Flow:

GRAVFØ-2





5.1.2.9 Subroutine INTEG

Purpose: To numerically integrate the equations of motion
(and the variational equations if desired) over
an integration step.

Input:

- o initial state deviations from conic $\underline{r}_o, \underline{v}_o$
- o perturbing accelerations, \underline{a}_o
- o state transition matrix at start, $\begin{bmatrix} A^o \\ B^o \\ C^o \\ D^o \end{bmatrix}$
- o gravity gradient matrix, $[G]$
- o current spacecraft mass, m
- o exhaust velocity, c
- o propulsive efficiency, η
- o step size, h
- o true state, \vec{R}, \vec{V}
- o covariance integration flag
- o augmented state covariance $[P]$
- o augmented state covariance derivative $[\dot{P}]$
- o Thrust controls
- o thrust acceleration, \vec{T}
- o dimension of state transition matrix, n
- o thrust transformation matrix $[F]$
- o current time, t
- o osculating spacecraft state $\underline{r}_{osc}, \underline{v}_{osc}$
- o mass variance, σ_M^2
- o acceleration proportionality variance, a_p
- o correlation time, τ
- o acceleration scale factor, a_s
- o acceleration resolution variance, a_r

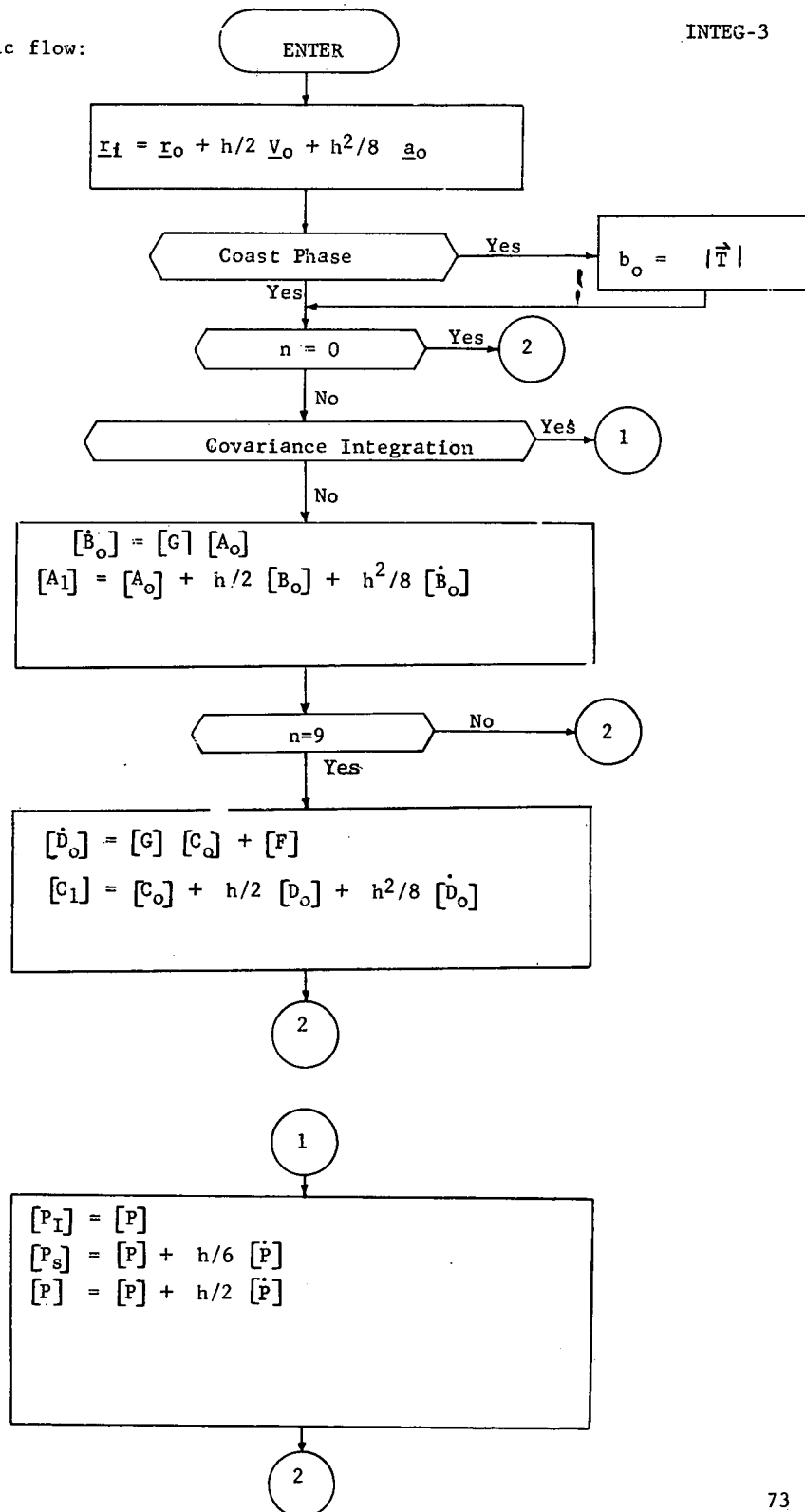
- Output;
- o integrated state deviations $\underline{r}_f, \underline{v}_f$
 - o updated true state, R, V
 - o mass variance, σ_M^2
 - o updated state transition matrix partitions $\begin{bmatrix} A_f \\ B_f \\ C_f \\ D_f \end{bmatrix}$
 - o updated augmented covariance matrix, $[P]$
 - o current time, t

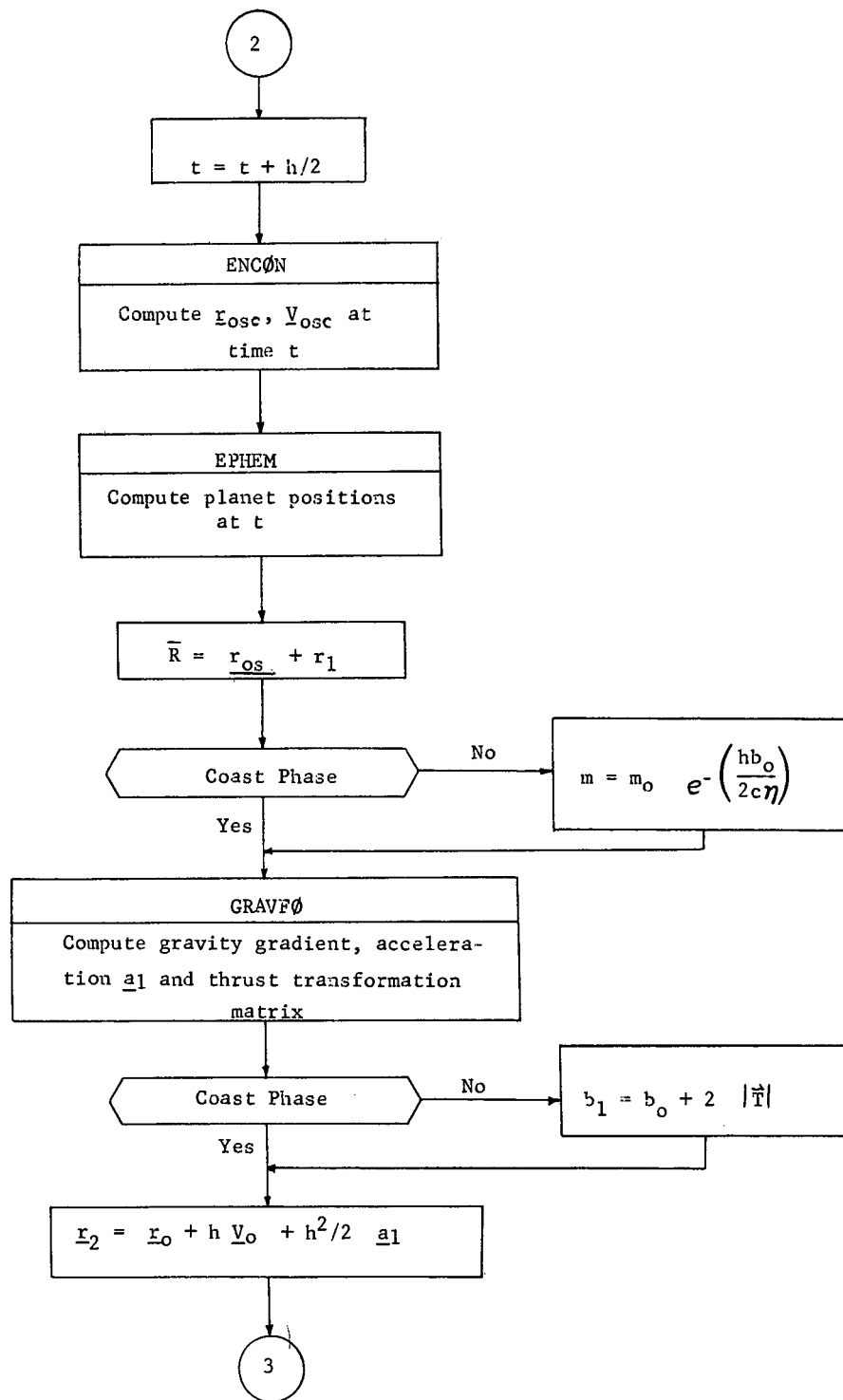
Remarks:

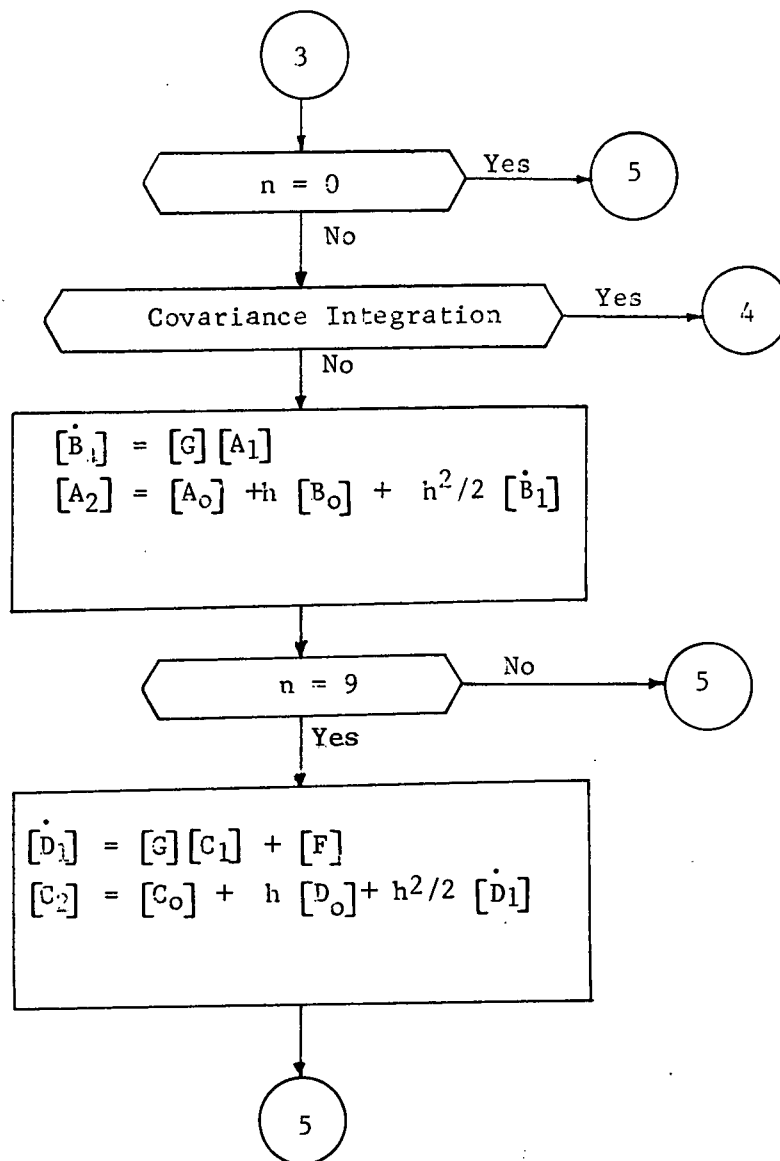
The numerical integration technique is a fourth order Nystrom.

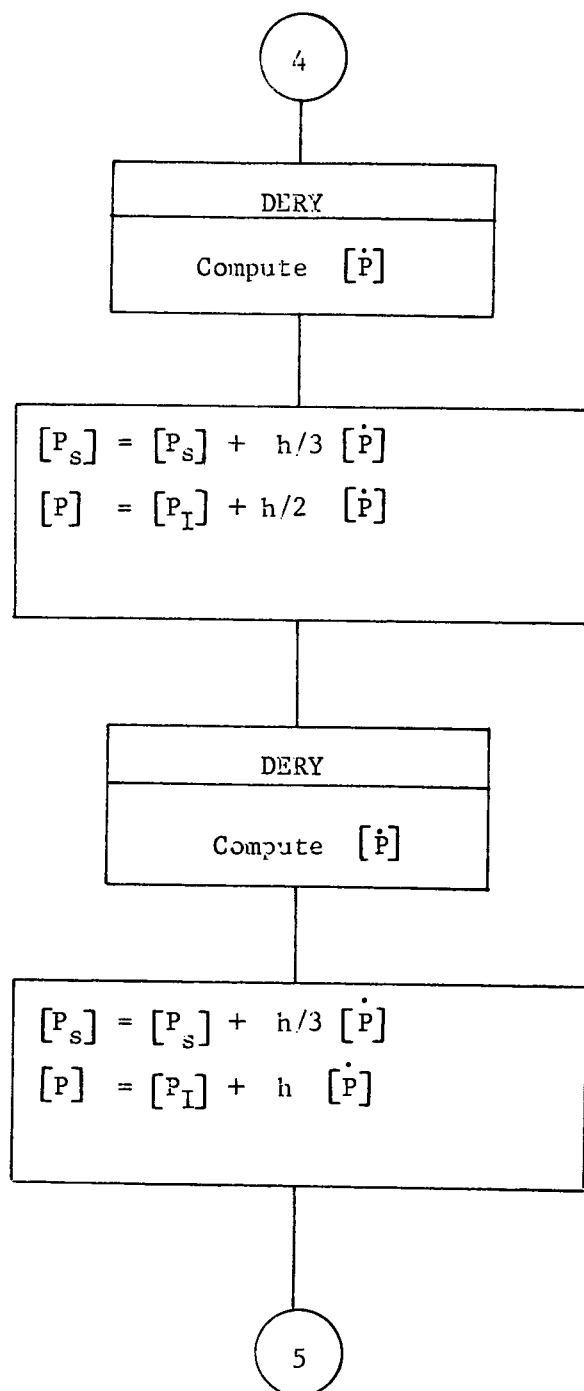
Logic flow:

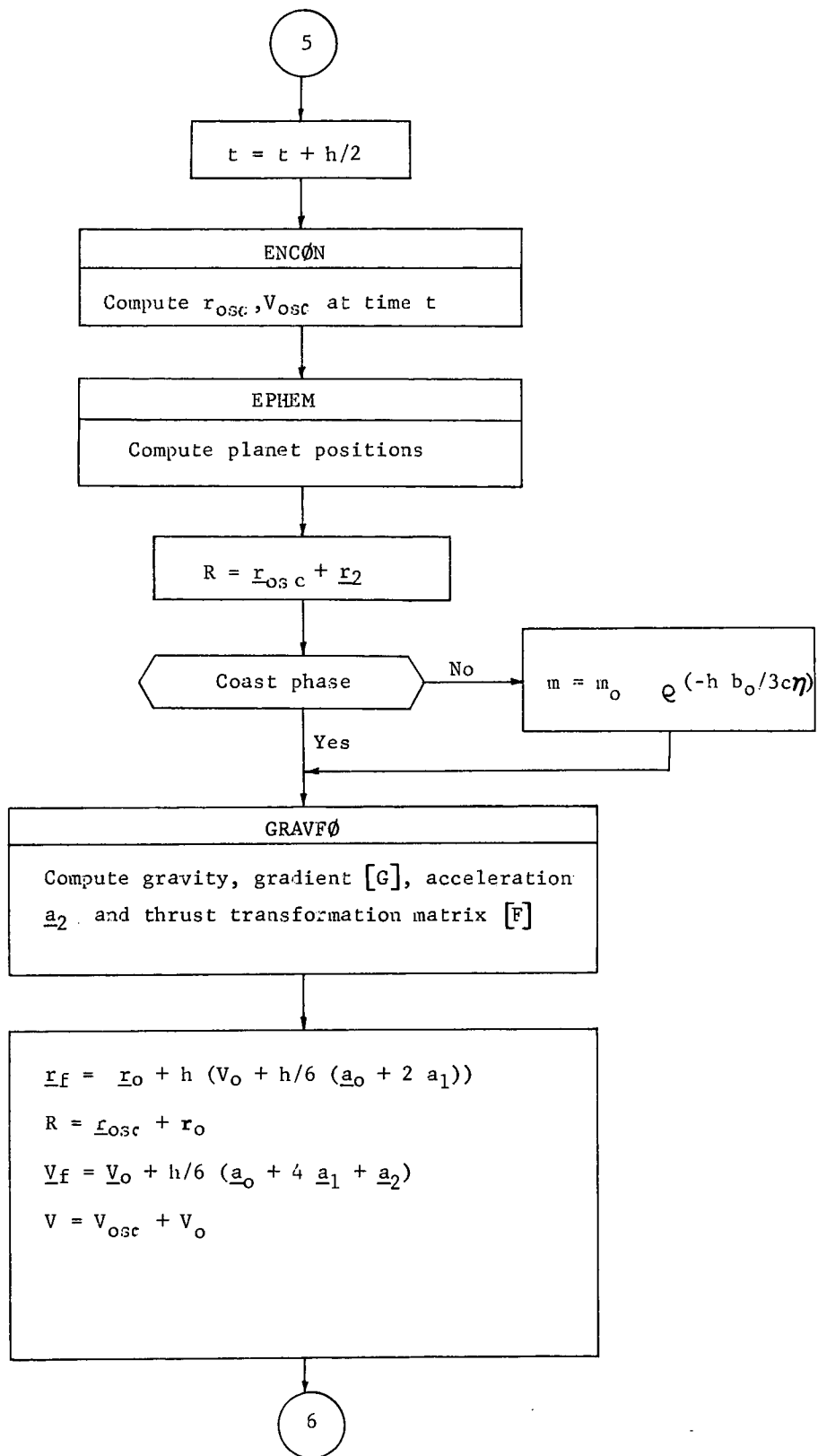
INTEG-3

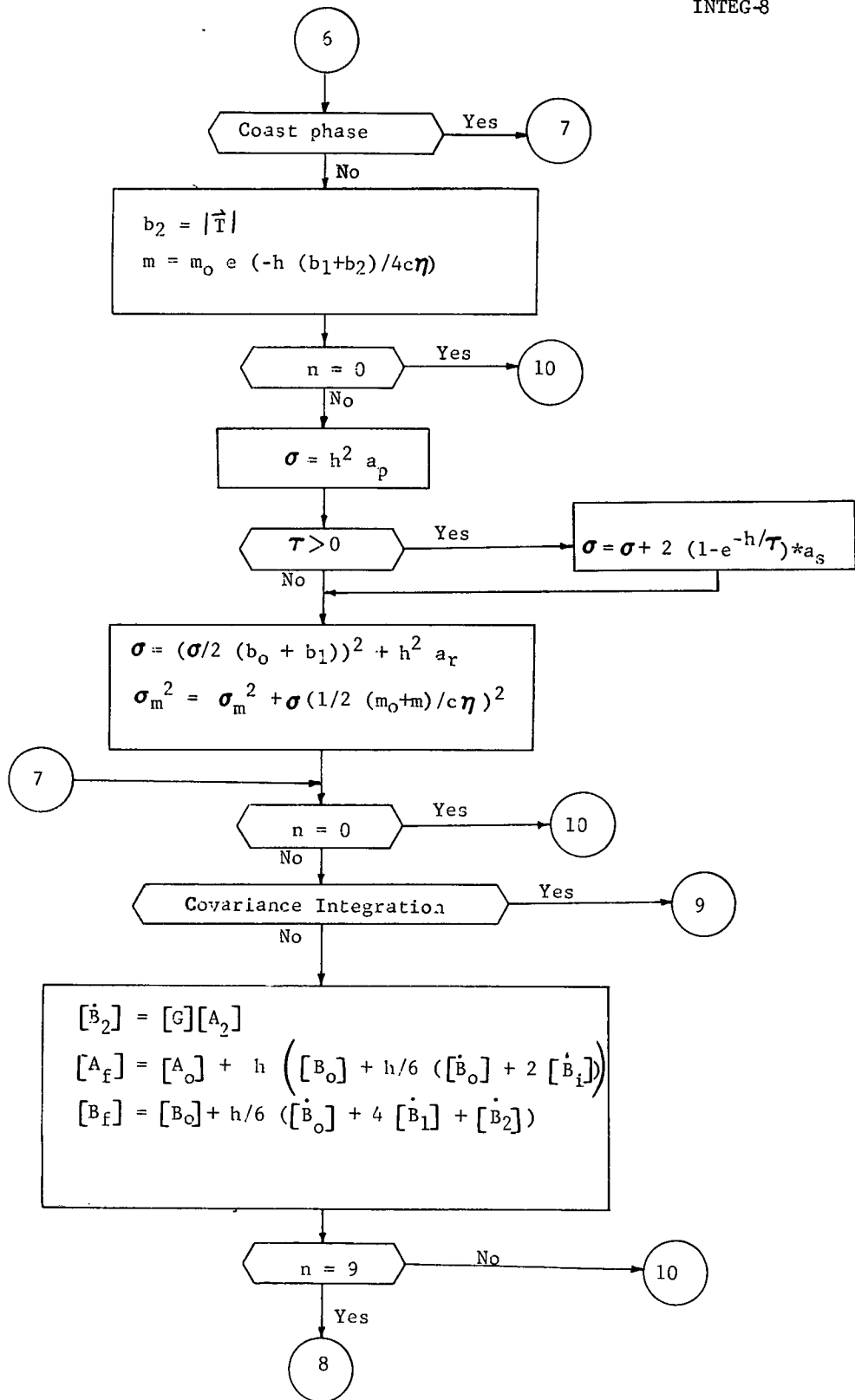


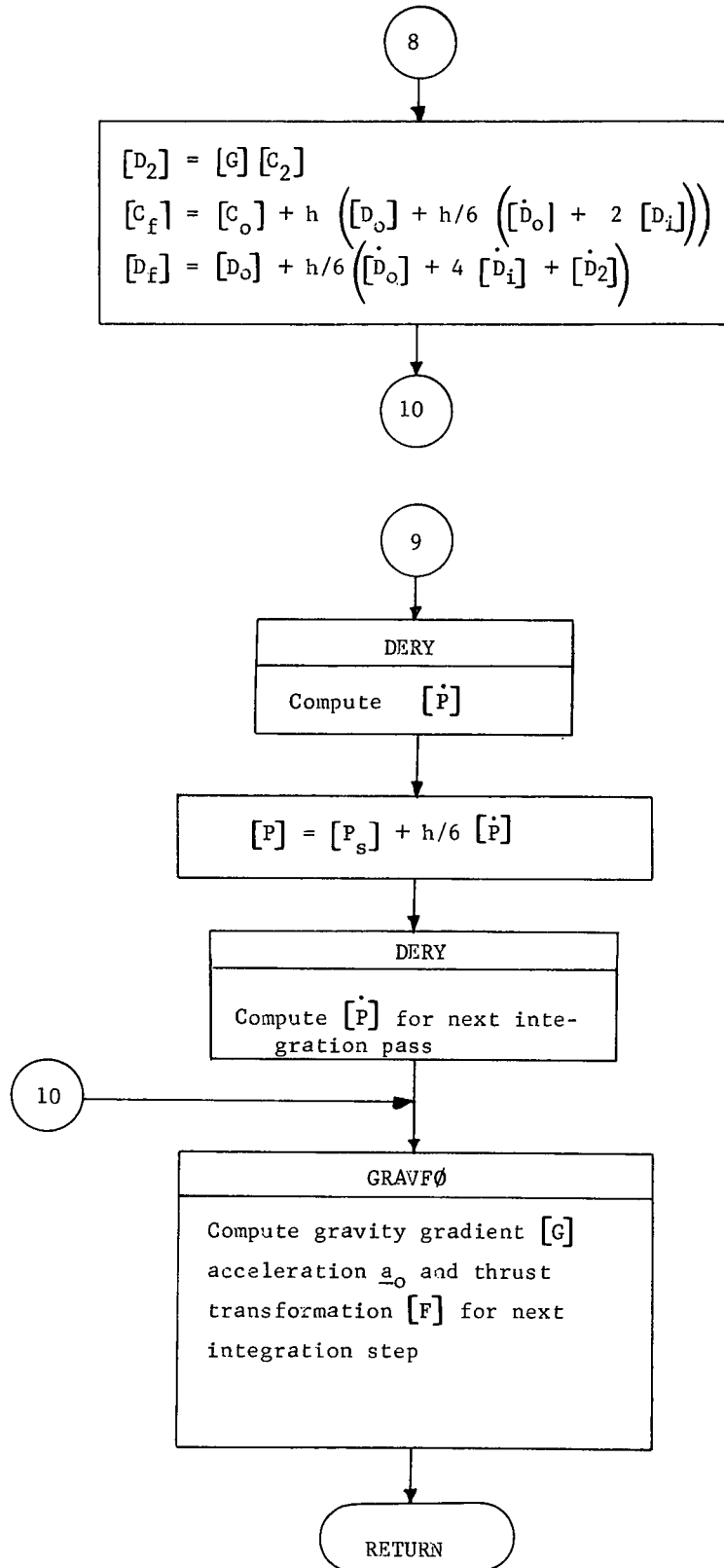












5.1.2.10 Function POWER

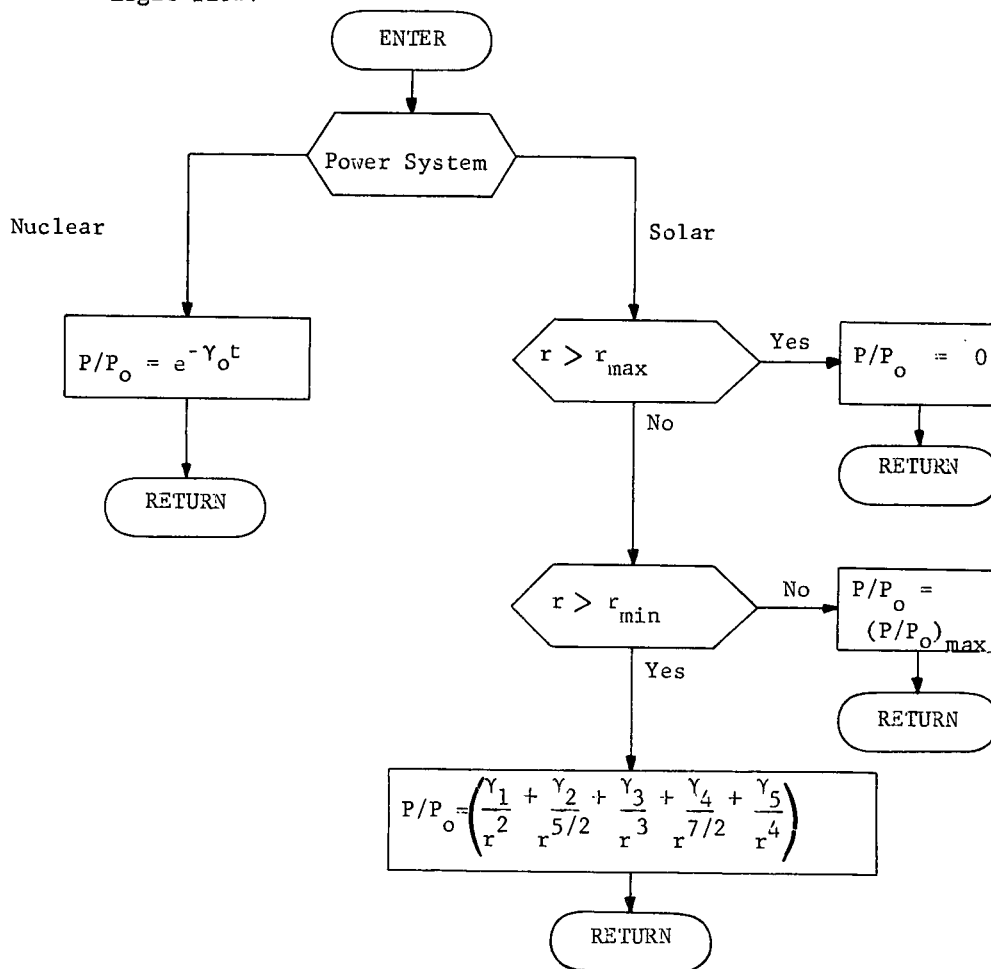
Purpose: To compute the power ratio available to the propulsion system.

Input;

- model selection
- flight time, t
- heliocentric distance (for solar propulsion), r
- power constants, γ_i
- range of usefulness for solar array,
 $r_{\min}, r_{\max}, (P/P_o)_{\max}$

Output: • power ratio, P/P_o

Logic flow:



5.2 Targeting Optimization Mode

5.2.1 Subroutine BUCKET

Purpose: To sort a set of independent elements in ascending order and to find a bounded minimum from the associated set of dependent elements.

Input:

- o set of independent elements, X_i
- o set of dependent elements, Y_i
- o number of elements, N

Output:

- o ordered set of independent elements, X_j
- o ordered set of dependent elements, Y_j
- o pointer, k, to a minimum dependent element

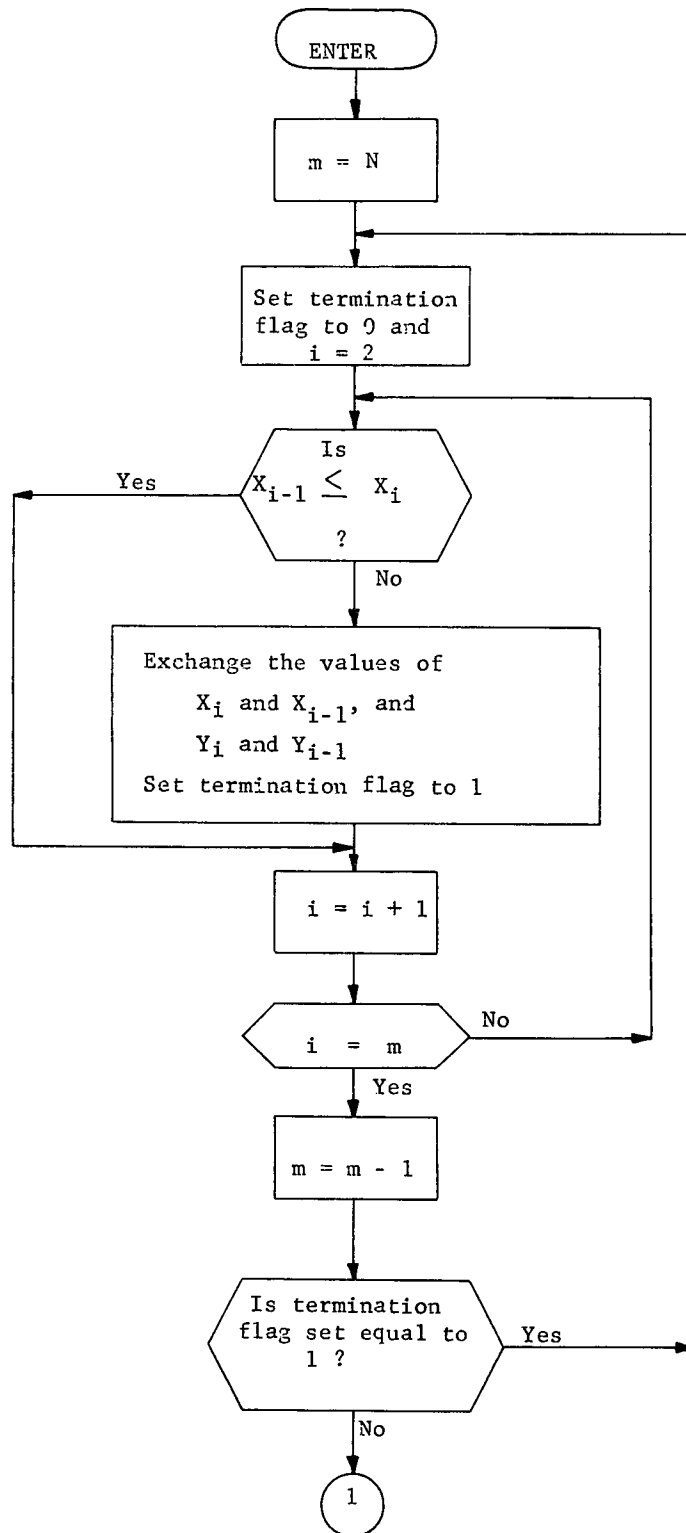
Remarks:

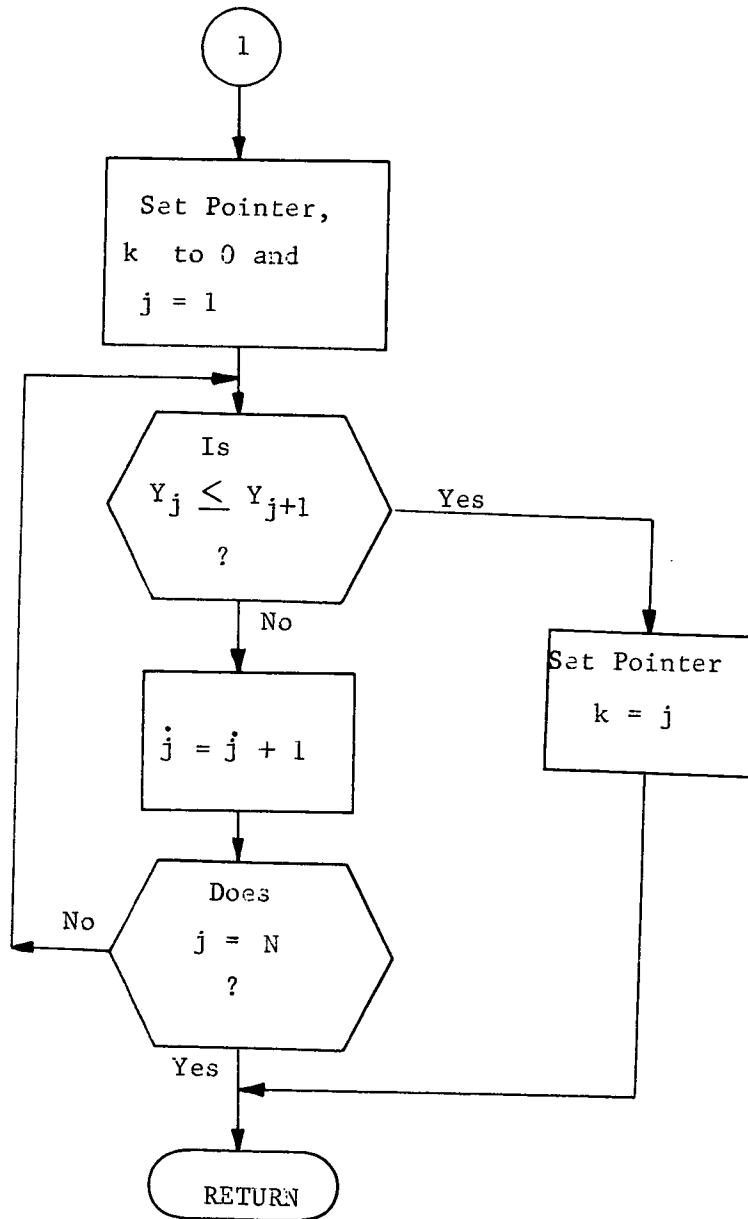
This routine is used in preparation for the polynomial curve fitting routine, MINMUM, to aid in calculating the new control profile.

BUCKET sorts pairs of elements (X_i, Y_i) in ascending order of the elements X_i and locates the element Y_k from the newly ordered pairs such that

$$Y_{k-1} < Y_k < Y_{k+1}$$

If this condition cannot be satisfied the pointer, K, is set to zero to indicate that no bounded minimum exists.





5.2.2 Subroutine DELU

Purpose: To compute the control correction vector.

Input:

- o submode designation
- o sensitivity weighting matrix, W
- o target sensitivity matrix, S
- o performance gradient, \underline{G}
- o target errors, $\Delta \underline{T}$
- o current control vector, \underline{U}
- o estimated radius of region of linearity
- o number of controls, M
- o number of targets, N

Output:

- o complete control correction vector $\Delta \underline{U}$
- o optimization control correction vector, $\Delta \underline{U}_1$
- o constraint control correction vector, $\Delta \underline{U}_2$

Remarks:

Subroutine DELU applies the projected gradient algorithm to compute the control correction vector, $\Delta \underline{U}$. The direction of the correction vector is dependent upon the submode designation. For example,

Targeting only: $\Delta \underline{U} = \Delta \underline{U}_2$

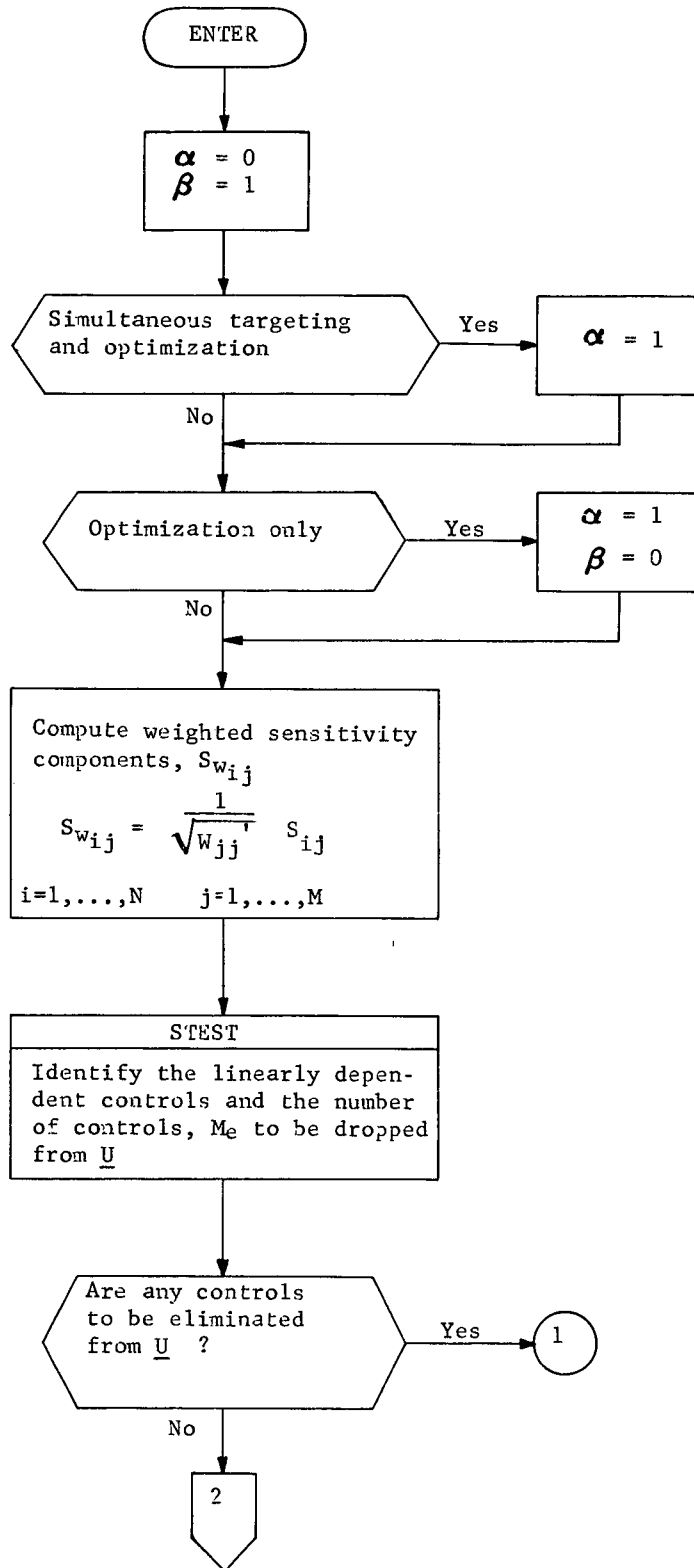
Targeting and optimization: $\Delta \underline{U} = \Delta \underline{U}_1 + \Delta \underline{U}_2$

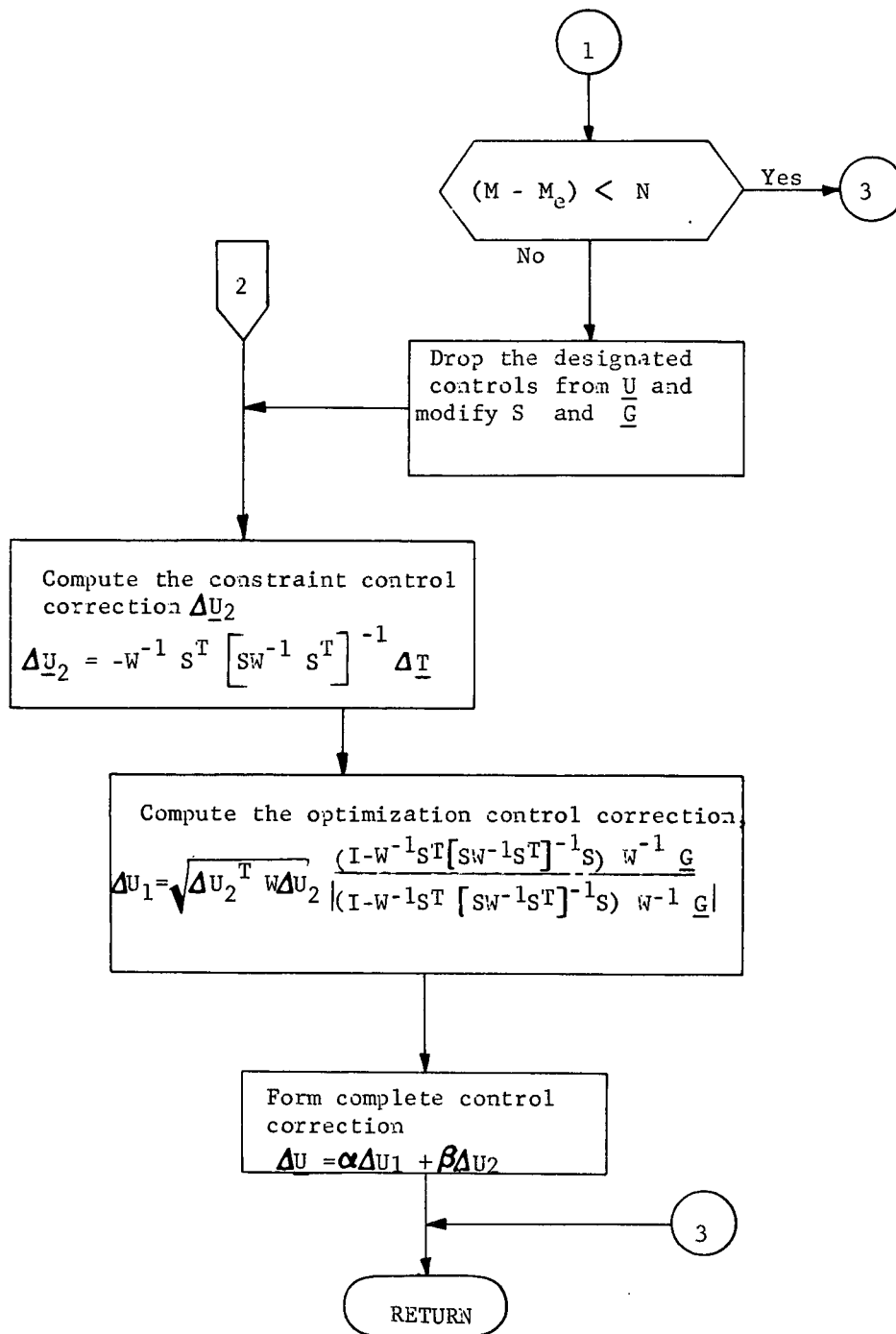
Optimization only: $\Delta \underline{U} = \Delta \underline{U}_1$

Linearly dependent controls are identified in subroutine STEST and are dropped from the subsequent matrix operations. No change is allowed in the omitted controls for the current iteration.

Logic flow:

DELU- 2





5.2.3 Subroutine FECS

Purpose: To calculate the performance index, the error index, the targeting sensitivity matrix, and the performance gradient.

Input:

- o desired target values, \underline{T}_o
- o number of targets, N
- o nominal controls, \underline{U}
- o number of controls, M
- o control perturbations, $\delta \underline{U}$
- o flag to indicate desired computations
 - oo generate nominal trajectory only
 - oo compute performance gradient \underline{G} and target sensitivity matrix S only
 - oo generate nominal trajectory, \underline{G} and S

Output:

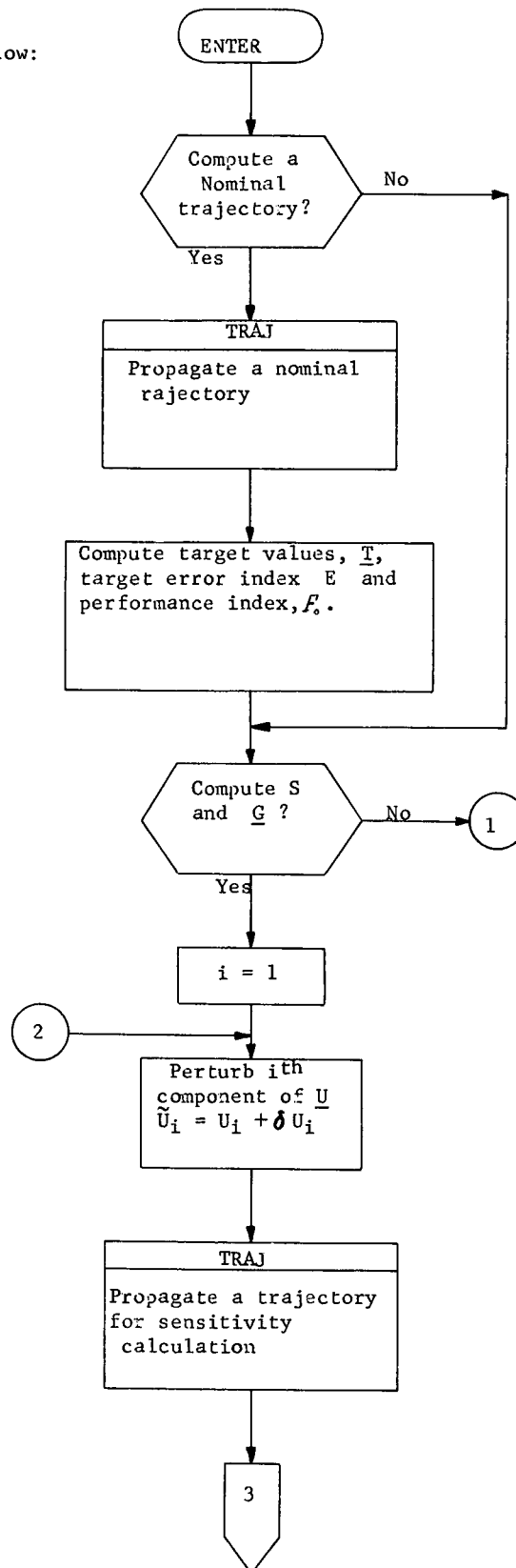
- o performance index, F
- o target error index, E
- o values of target parameters \underline{T} for nominal trajectory
- o performance gradient, \underline{G}
- o targeting sensitivity matrix, S

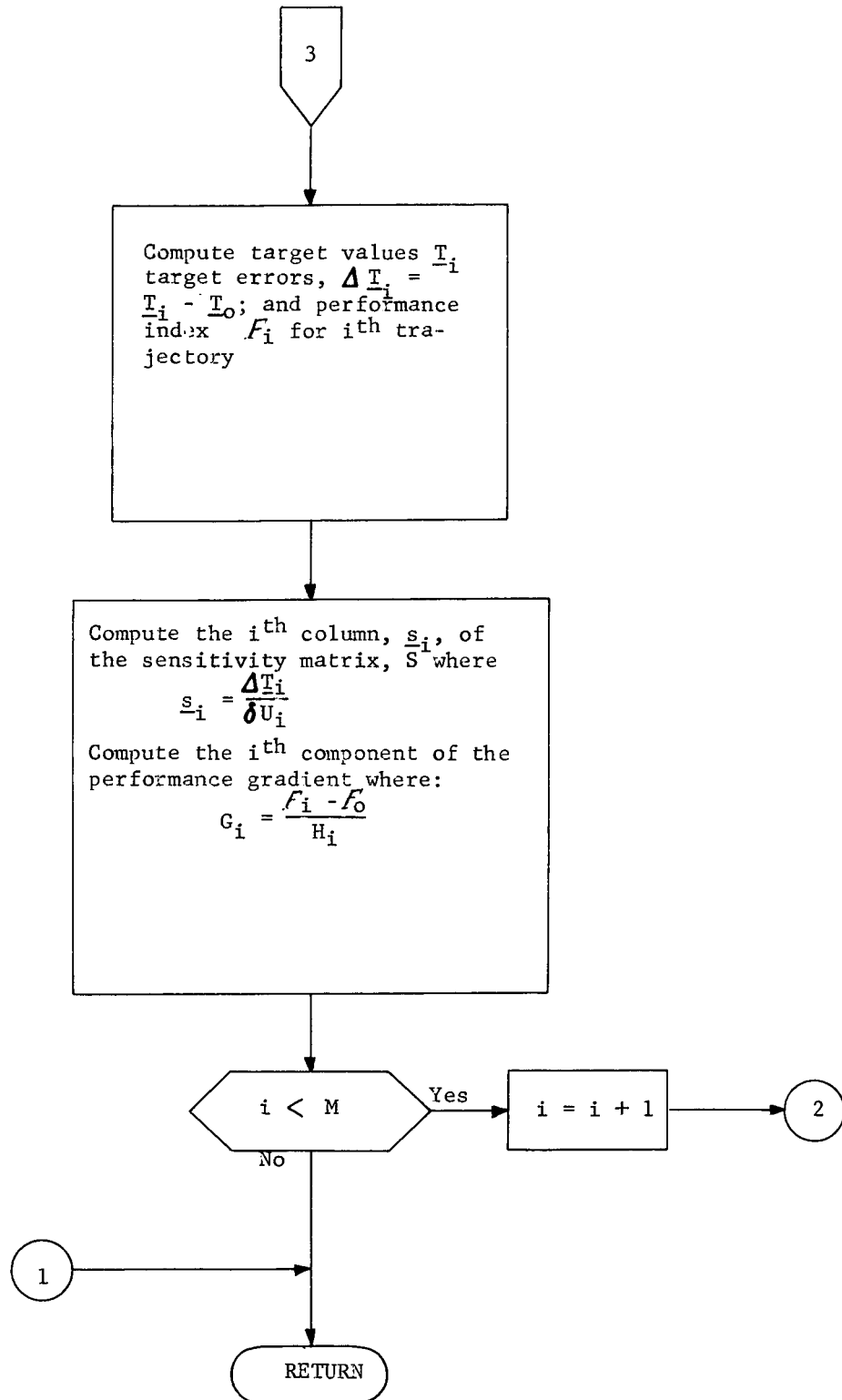
Remarks:

The performance and target error indices which are computed in FECS are used in subroutine TEST (section 5.2.11) to determine the routine submode for the next iteration. The performance index is simply the final spacecraft mass and the error index is the sum of the squares of the target error.

Logic flow:

FECS-2





5.2.4 Subroutine FUNCT

Purpose: To calculate the net cost-function for a trial trajectory.

Input:

- o current control vector, \underline{U}
- o trial control change scale factor, γ
- o control change vector, $\Delta \underline{U}$
- o current performance index, F_o
- o current sensitivity matrix, S
- o current performance gradient, \underline{G}
- o desired target values, \underline{T}_o
- o submode designation
 - oo targeting only
 - oo targeting and optimization
 - oo optimization only

Output: o net cost-function value, $F(\gamma)$ for the trial trajectory

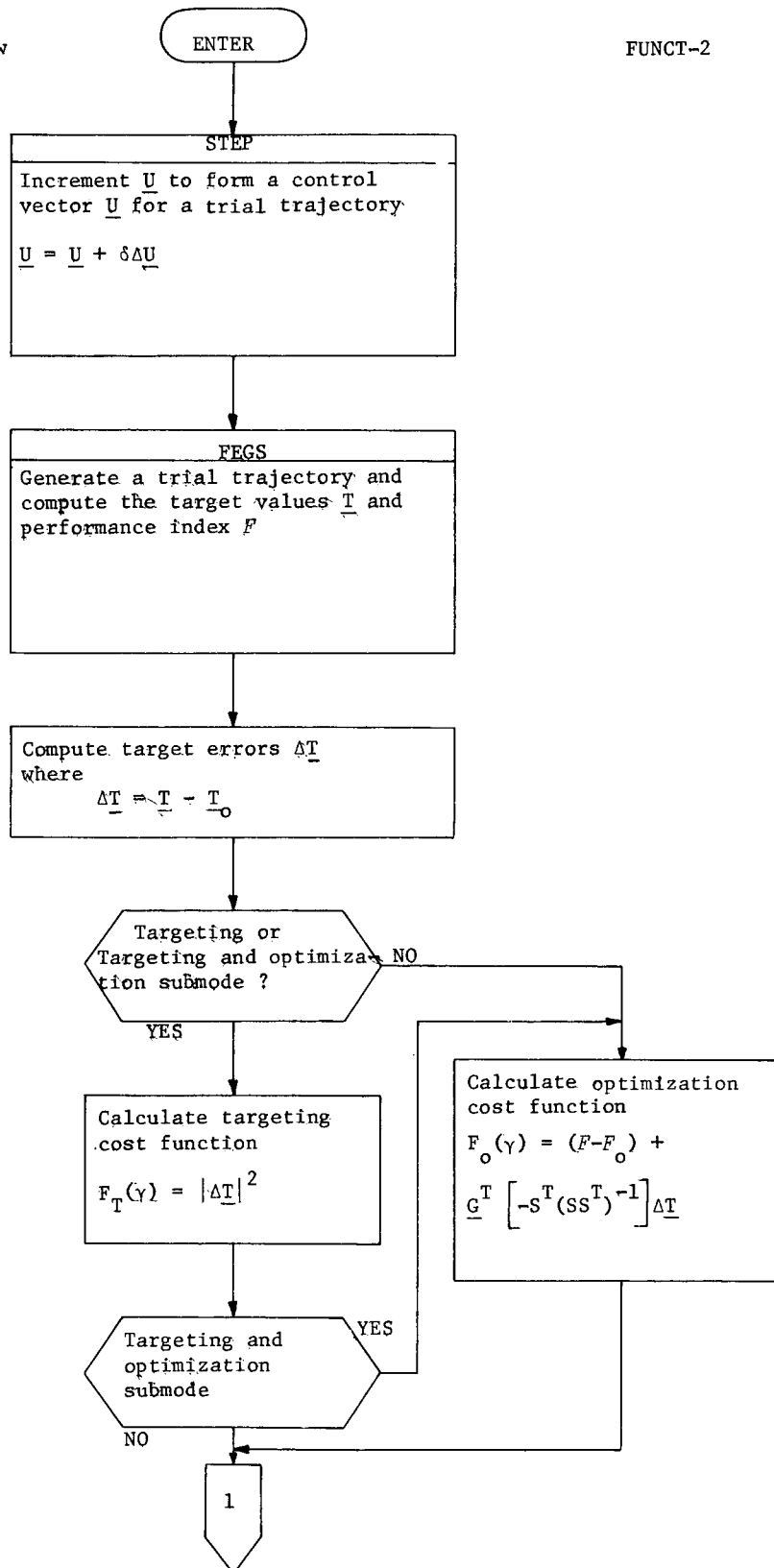
Remarks:

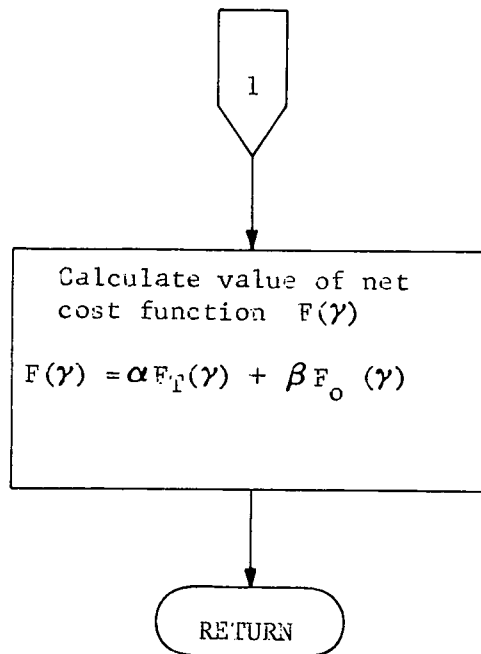
The net cost-function is described in Section 5.2.8 (Subroutine SIZE).

$$F(\gamma) = \alpha F_T(\gamma) + \beta F_o(\gamma)$$

$$\alpha = \begin{cases} 1 & \text{for targeting only or simultaneous targeting and optimization} \\ 0 & \text{for optimization only} \end{cases}$$

$$\beta = \begin{cases} 1 & \text{for optimization or simultaneous targeting and optimization} \\ 0 & \text{for targeting only} \end{cases}$$





5.2.5 Subroutine GENMIN

Purpose: To generate a series of trial trajectories based on control change vectors of different magnitude and to choose the best control change scale factor.

Input:

- o current net cost-function value, $F(\gamma) \Big|_{\gamma=0}$
- o value of the first derivative of the net cost-function evaluated at $\gamma=0$, $F'(0)$
- o curve fitting tolerance for trail steps, η
- o maximum value of γ

Output:

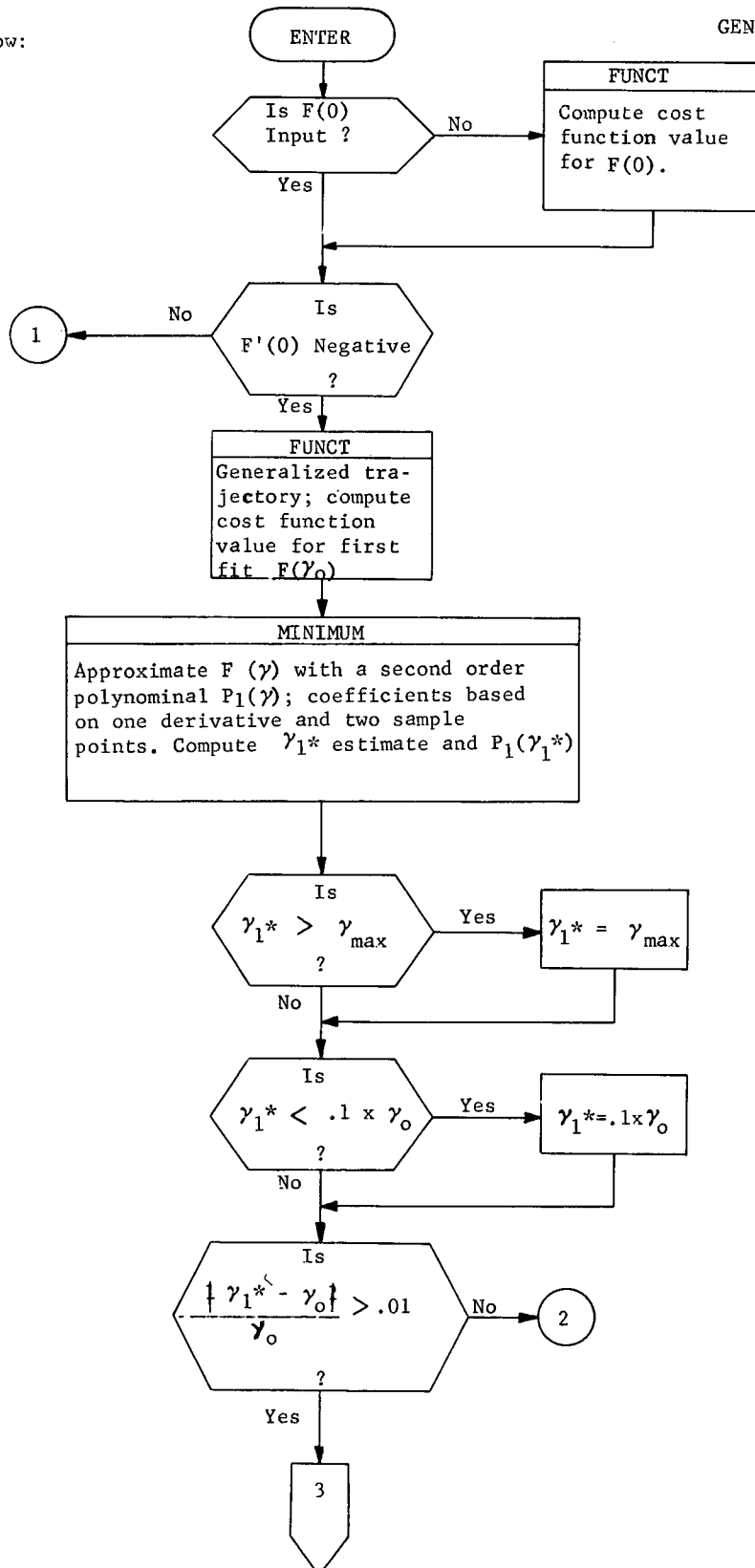
- o value of the net cost-function for each trial trajectory
- o minimum value of net cost function, $F(\gamma^*)$
- o minimizing scale factor, γ^*

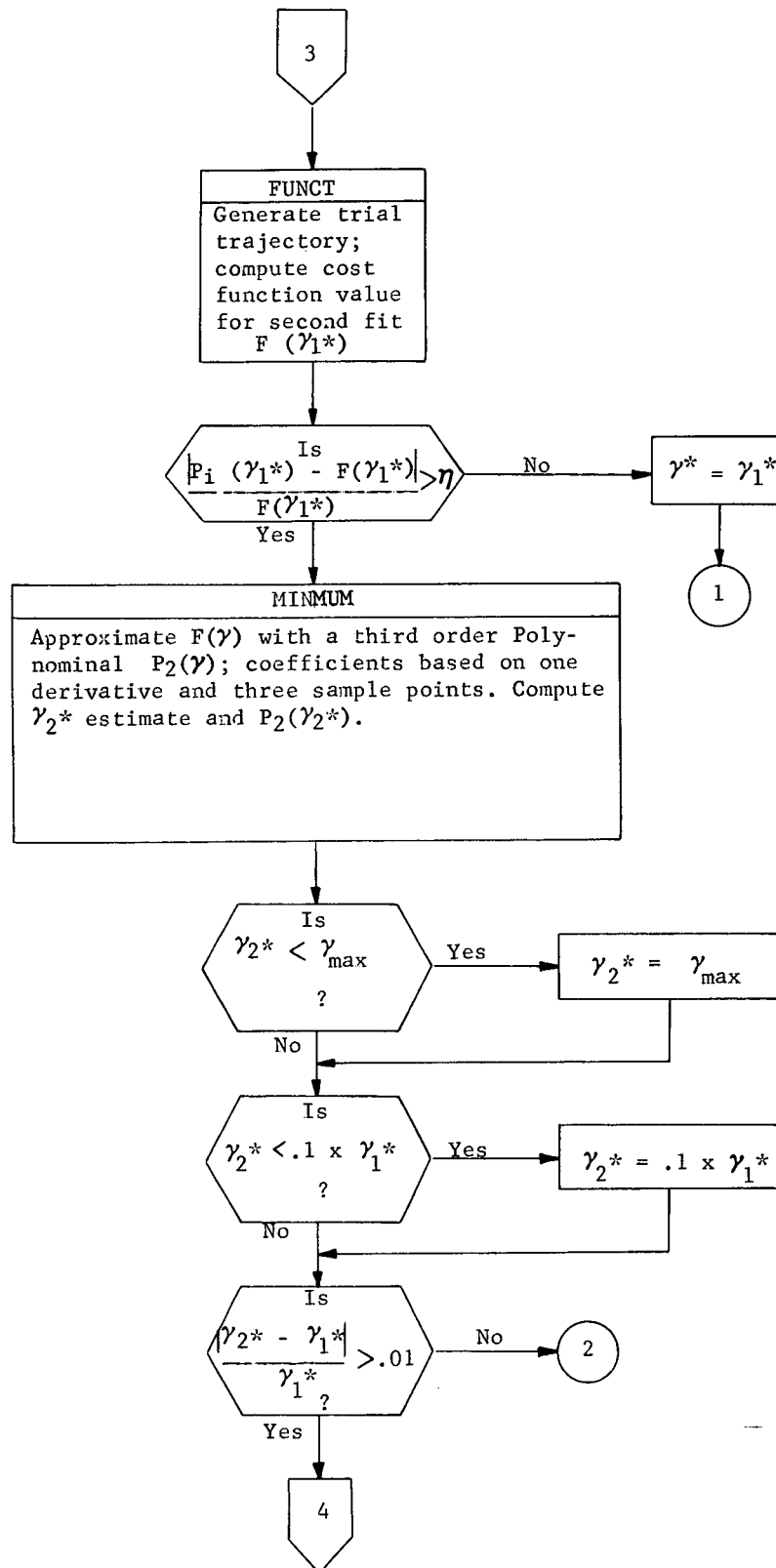
Remarks:

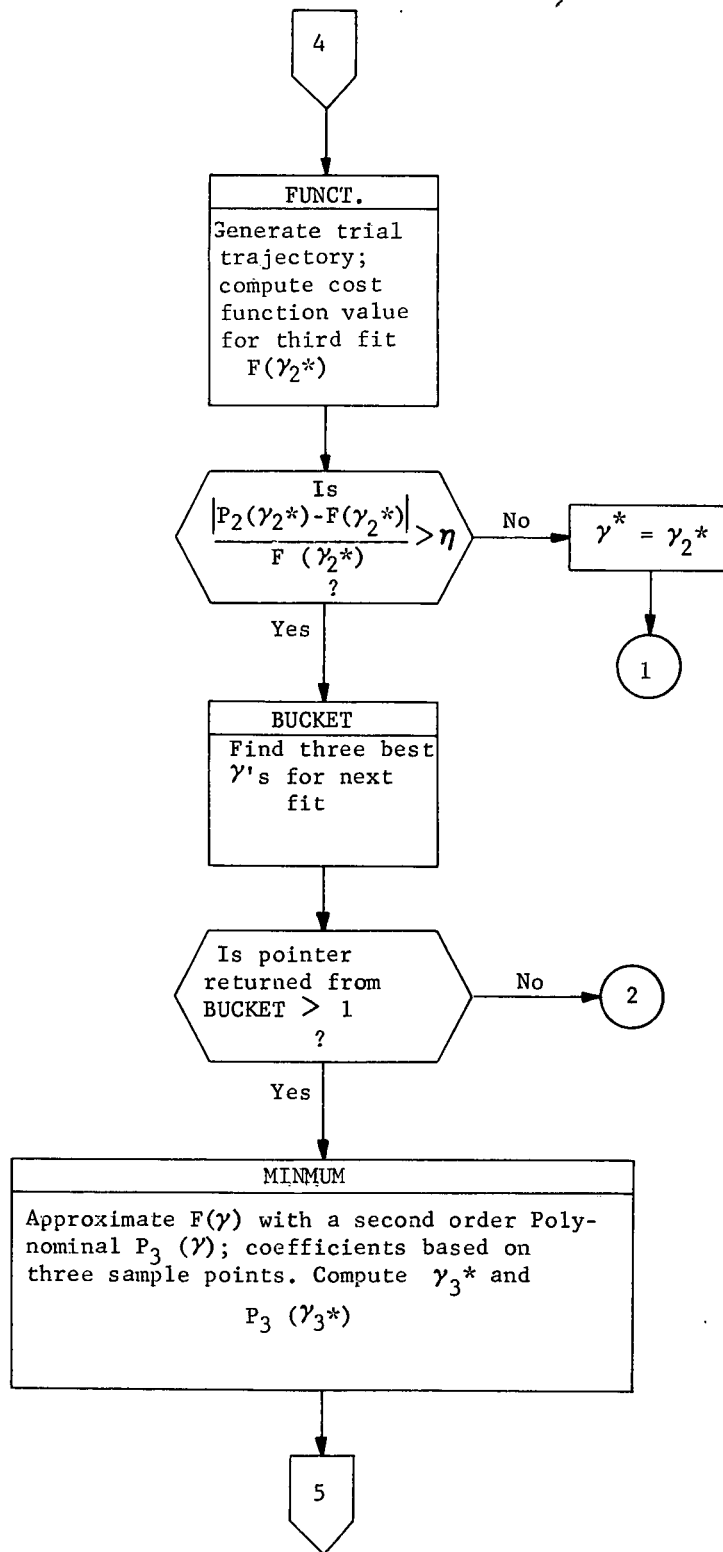
The net cost-function is described in Section 5.2.8 (SIZE)

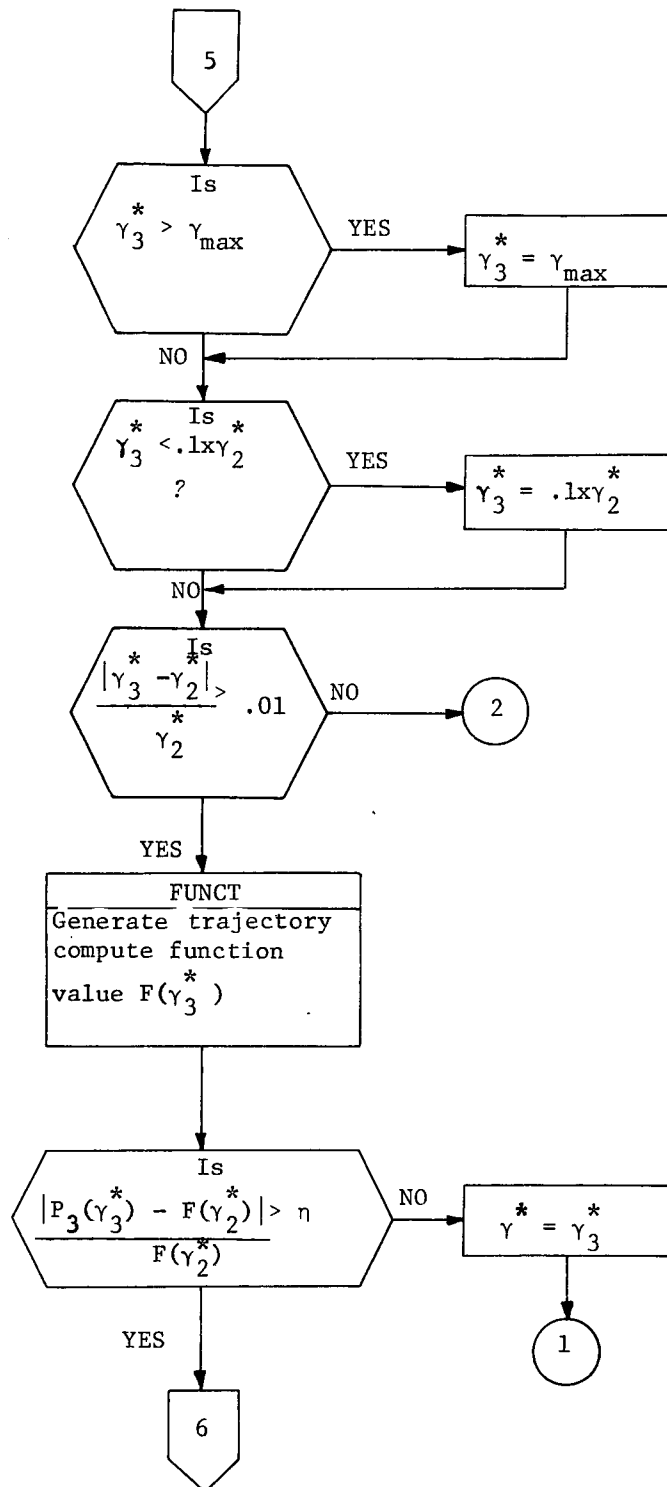
Logic flow:

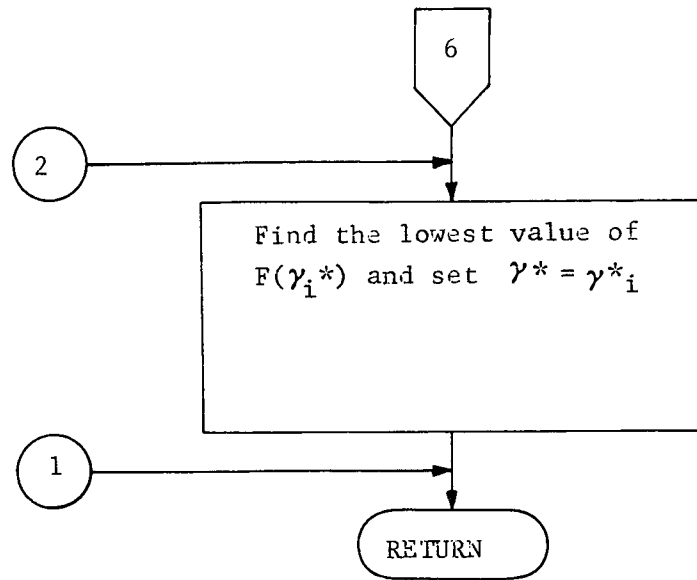
GENMIN-2











5.2.6 Subroutine GRID

Purpose: To generate a family of trajectories.

Input:

- o nominal controls, \underline{U}
- o control increment, $\Delta \underline{U}$
- o number of controls, M
- o maximum value of scale factor γ_{\max} for control increment
- o desired target values
- o flag designating two incremented controls per grid trajectory

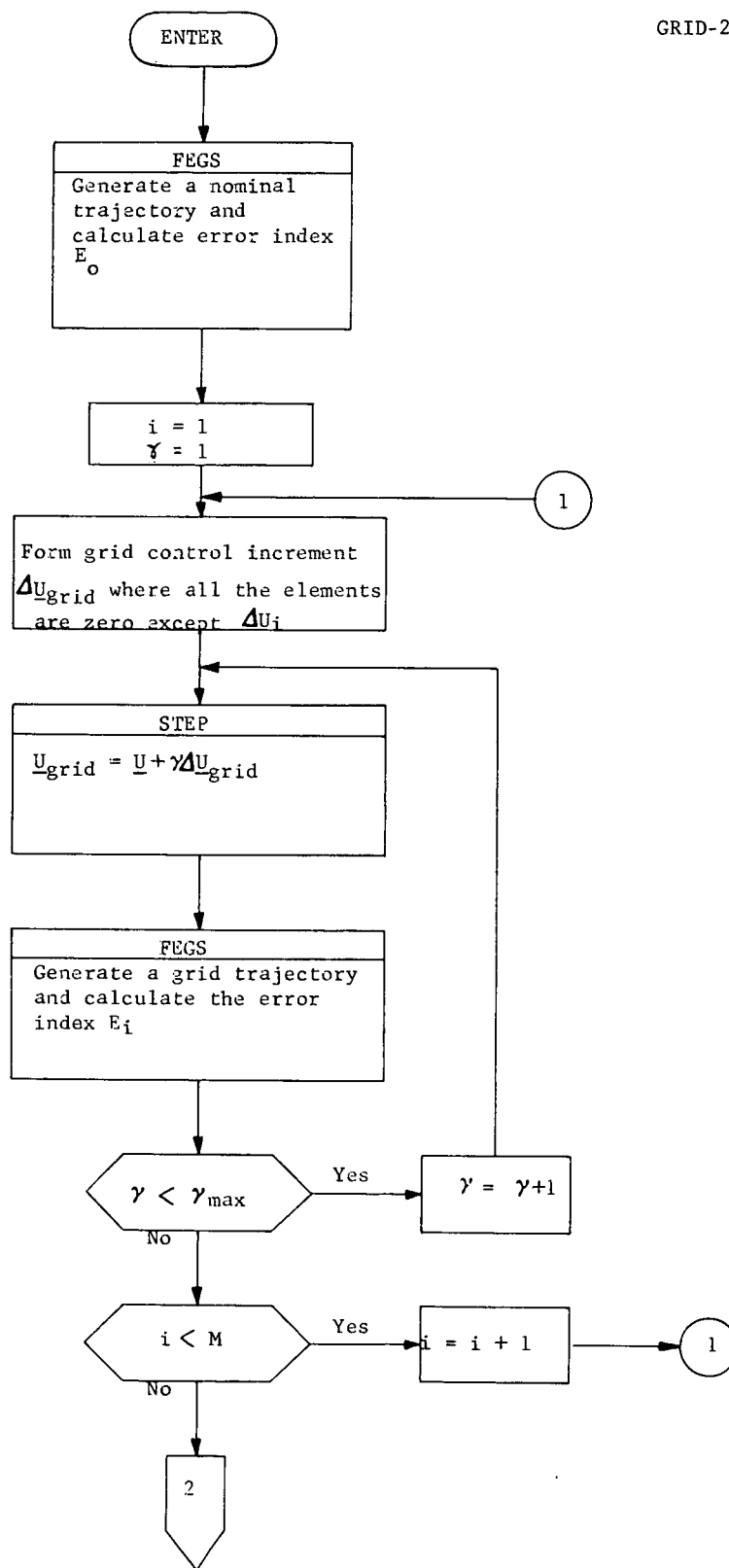
Output: o External

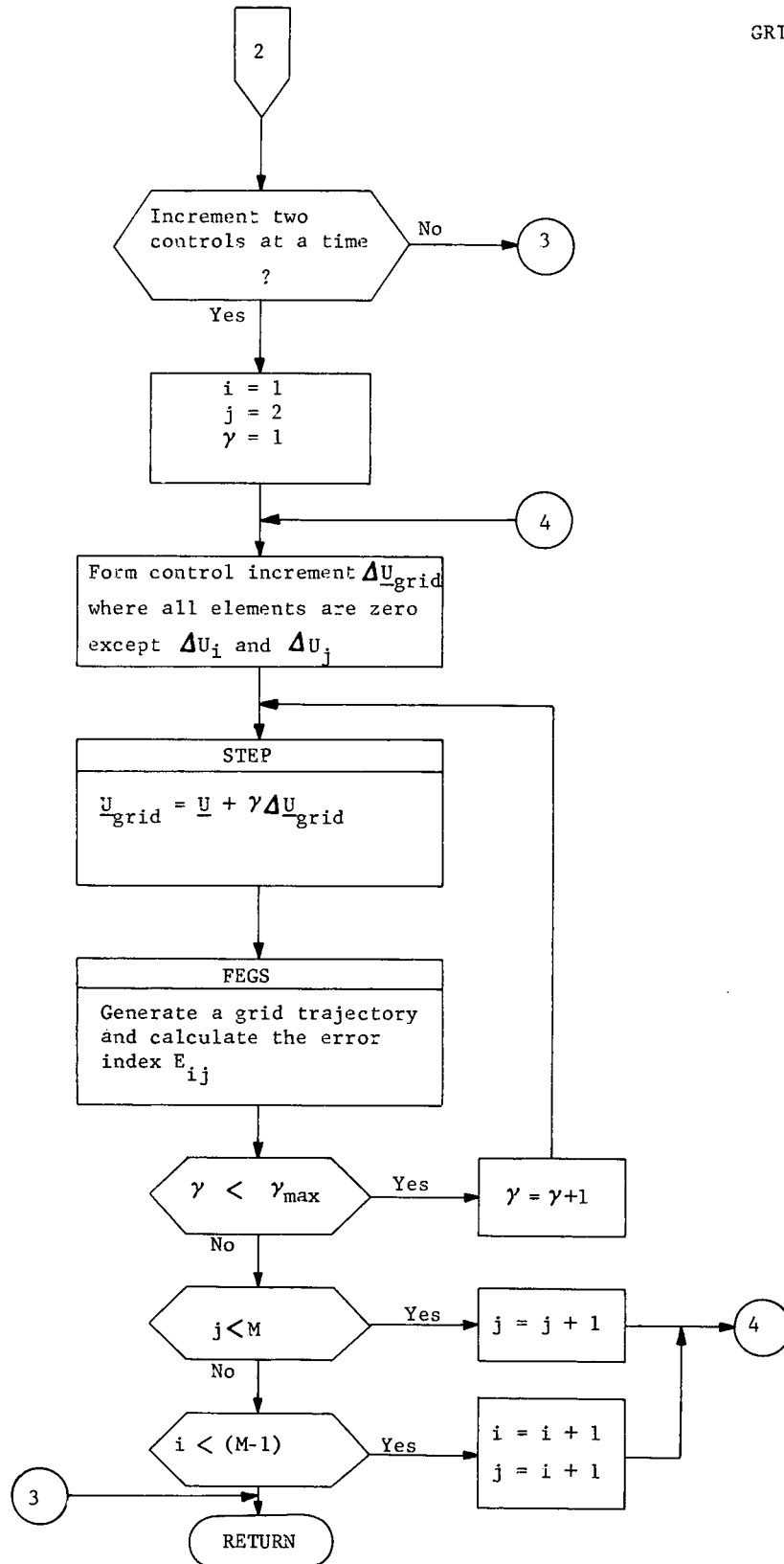
Remarks:

Subroutine GRID is used to generate a grid of trajectory target error indices. The error indices are used either to direct a finer grid search or to choose a control profile to enter the other submodes of TOM.

Logic flow:

GRID-2





5.2.7 Subroutine MINMUM

Purpose: To find the minimum value of a function, $F(\gamma)$, and the minimizing independent variable, γ^*

Input:

- o flag denoting type of polynomial approximation
 - oo second order polynomial, coefficients based on two sample points and one derivative evaluated at a point
 - oo third order polynomial, coefficients based on three sample points and one derivative evaluated at a point
 - oo second order polynomial, coefficients based on three sample points
- o set of at most three distinct values of the function, $F(\gamma)$
- o set of corresponding independent variable values, γ
- o value of the first derivative of the function $F(\gamma)$ evaluated at $\gamma=0$

Output:

- o estimate of the minimum value of the function, $F(\gamma^*)$
- o value of the minimizing parameter γ^*

Remarks:

The function, $F(\gamma)$, is approximated by either a second or third order polynomial, $P(\gamma)$, in order to compute analytically the minimizing parameter γ^* . The polynomial approximation is of the form

$$F(\gamma) \cong P(\gamma) = \sum_{i=0}^n a_i \gamma^i$$

where $n=2$ or $n=3$. The following three cases describe the method of approximation and the resulting minimization process.

Case 1

F is fitted with a quadratic polynomial based on:

$$1) F(0)$$

$$2) F'(0) = \left. \frac{dF(\gamma)}{d\gamma} \right|_{\gamma=0}$$

$$3) F(\gamma_0) \text{ where } \gamma_0 > 0 \text{ is an initial estimate of } \gamma^*$$

The quadratic polynomial coefficients are calculated from the formulae

$$a_0 = F(0)$$

$$a_1 = F'(0)$$

$$a_2 = F(\gamma_0) - \frac{a_0}{\gamma_0^2} + \frac{a_1}{\gamma_0}$$

The independent variable value minimizing the quadratic is

$$\gamma^* = -\frac{a_1}{2a_2}$$

Case 2

F is fitted with a cubic polynomial based on:

$$1) F(0)$$

$$2) F'(0)$$

$$3) F(\gamma_0) \text{ where } \gamma_0 \text{ is as in Case 1}$$

$$4) F(\gamma_1) \text{ where } \gamma_1 > 0 \text{ is a sample value}$$

The cubic polynomial coefficients are calculated from the following formulae

$$a_0 = F(0)$$

$$a_1 = F'(0)$$

$$a_2 = \frac{F(\alpha\lambda) - \alpha^3 F(\lambda) - \lambda\alpha(1+\alpha)a_1 - (1+\alpha+\alpha^2)a_0}{(1-\alpha)}$$

$$a_3 = \frac{\lambda a_1 \alpha + a_0(1+\alpha) + \frac{F(\lambda) - F(\alpha\lambda)}{\lambda\alpha}}{\lambda^3 \alpha^2}$$

where $\lambda = \max (\gamma_0, \gamma_1)$
 $= \min (\lambda_0, \gamma_1) / \lambda$

The independent variable value, γ^* , minimizing P is

$$\gamma^* = \frac{(-a_2 + a_2^2 - 3a_3 a_1)}{3a_3}$$

Case 3 A quadratic polynomial is fitted to $F(\gamma_2)$, $F(\gamma_3)$ and $F(\gamma_4)$ where γ_2, γ_3 , and γ_4 are greater than zero and represent sample values of γ .

It is assumed that the input of satisfy two conditions

- 1) $\gamma_2 < \gamma_3 < \gamma_4$
- 2) $F(\gamma_2) < F(\gamma_3) < F(\gamma_4)$

The formulae for the quadratic coefficients are as follows:

$$b_{ij} = \gamma_i \gamma_j$$

$$c_{ij} = \gamma_i + \gamma_j$$

$$d_{ij} = \gamma_i - \gamma_j$$

$$a_0 = \frac{b_{34}}{d_{23} d_{24}} F(\gamma_2) + \frac{b_{24}}{d_{32} d_{34}} F(\gamma_3) + \frac{b_{23}}{d_{42} d_{43}} F(\gamma_4)$$

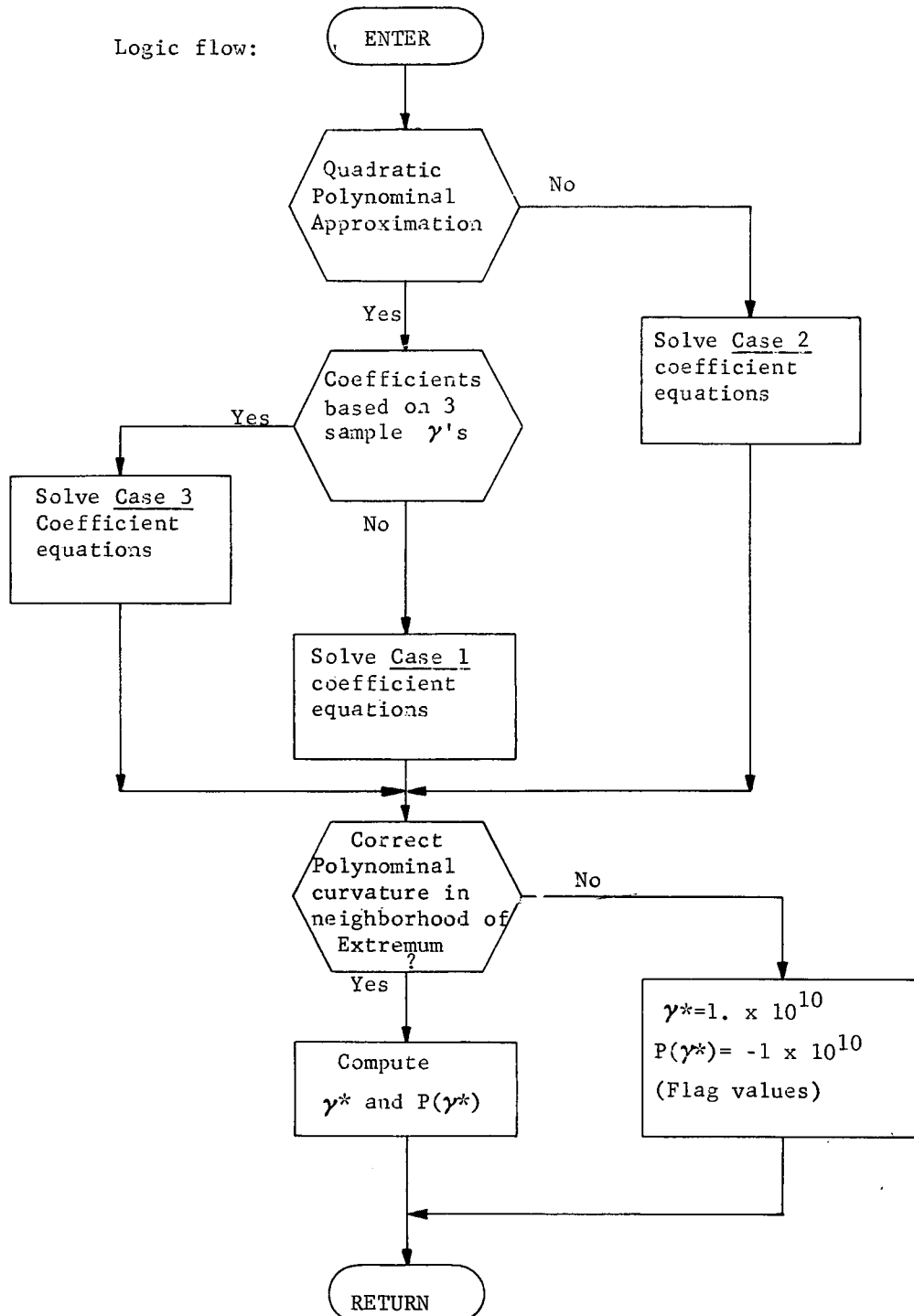
$$a_1 = - \frac{c_{34}}{d_{23} d_{24}} F(\gamma_2) - \frac{c_{24}}{d_{32} d_{34}} F(\gamma_3) - \frac{c_{23}}{d_{42} d_{43}} F(\gamma_4)$$

$$a_2 = \frac{F(\gamma_2)}{d_{23} d_{24}} + \frac{F(\gamma_3)}{d_{32} d_{34}} + \frac{F(\gamma_4)}{d_{42} d_{43}}$$

The independent variable value is the same as in Case 1.

$$\gamma^* = - \frac{a_1}{2a_2}$$

Logic flow:



5.2.8 Subroutine SIZE

- Purpose: To calculate the magnitude of the control change vector.
- Input:
- o current control vector, \underline{U}
 - o value of the performance index, F
 - o percentage of target error to be corrected in one iteration
 - o values of target errors, $\Delta \underline{T}$
 - o target tolerances
 - o target sensitivities, S
 - o performance gradient, \underline{G}
 - o estimated size of region of linearity
 - o submode designation
 - o individual control scale factors
 - o type of weighting matrix to be computed
 - o initial estimate of control change scaling factor, γ
 - o curve fitting tolerance for trial steps, η
- Output:
- o complete control change vector, $\Delta \underline{U} = \Delta \underline{U}_1 + \Delta \underline{U}_2$
 - o optimization test angle, θ

Remarks:

Prior to calling subroutine DELU, the targeting error correction for the current iteration is computed. The nonlinear effects of certain targeting problems require that only a certain percentage of the target error is removed in any one step to prevent divergence.

For any particular control vector \underline{U} in the independent-variable (control) space the projected gradient algorithm reduces the multi-dimensional problem to a one dimensional search either along the constraint

direction to minimize the sum of the squares of the constraint violations or along the optimization direction to minimize the estimated net cost-function. In either case, once the initial control vector \underline{U} and the direction of search $\Delta\underline{U}$ are specified, the problem reduces to the numerical minimization of a function of a single variable - namely the scaling factor γ .

Subroutine GENMIN is called to compute the value of the scaling factor γ^* which minimizes a function $F(\gamma)$ in both the constraint direction ($\Delta\underline{U}_2$) and optimization direction ($\Delta\underline{U}_1$) or each direction individually depending on the submode designation. The net cost-function is the sum of two functions, $F_T(\gamma)$ and $F_O(\gamma)$.

$$F(\gamma) = \alpha F_T(\gamma) + \beta F_O(\gamma)$$

$$\alpha = \begin{cases} 1 & \text{for targeting only or simultaneous targeting and optimization} \\ 0 & \text{for optimization only} \end{cases}$$

$$\beta = \begin{cases} 1 & \text{for optimization or simultaneous targeting and optimization} \\ 0 & \text{for targeting only} \end{cases}$$

The first derivative $F'(\gamma) \Big|_{\gamma=0}$ is used in the one dimensional search to find γ^* and is calculated in SIZE prior to the call to GENMIN.

The function $F_T(\gamma)$ to be minimized along the constraint direction $\Delta\underline{U}_2$ is the sum of the squares of the target errors

$$F_T(\gamma) = ||\Delta\underline{T}(\underline{U} + \gamma \Delta\underline{U}_2)||^2$$

The first derivative evaluated at $\gamma=0$ is then

$$F'_T(0) = 2\Delta\underline{T}^T(\underline{U})S\Delta\underline{U}_2$$

The function $F_o(\gamma)$ to be minimized along the optimization direction ΔU_1 is the estimated net cost-function which is defined

$$F_o(\gamma) = \underbrace{F(U + \gamma \Delta U_1) - F(U)}_{\text{Change in performance index produced by a step of length } \gamma \text{ along } \Delta U_1} + \underbrace{G^T(U) \left[-S^T(SS^T)^{-1} \Delta T(U + \gamma \Delta U_1) \right]}_{\text{Linearized approximation to change in performance index required to maintain the current target errors.}}$$

Change in performance index produced by a step of length γ along ΔU_1 . Linearized approximation to change in performance index required to maintain the current target errors.

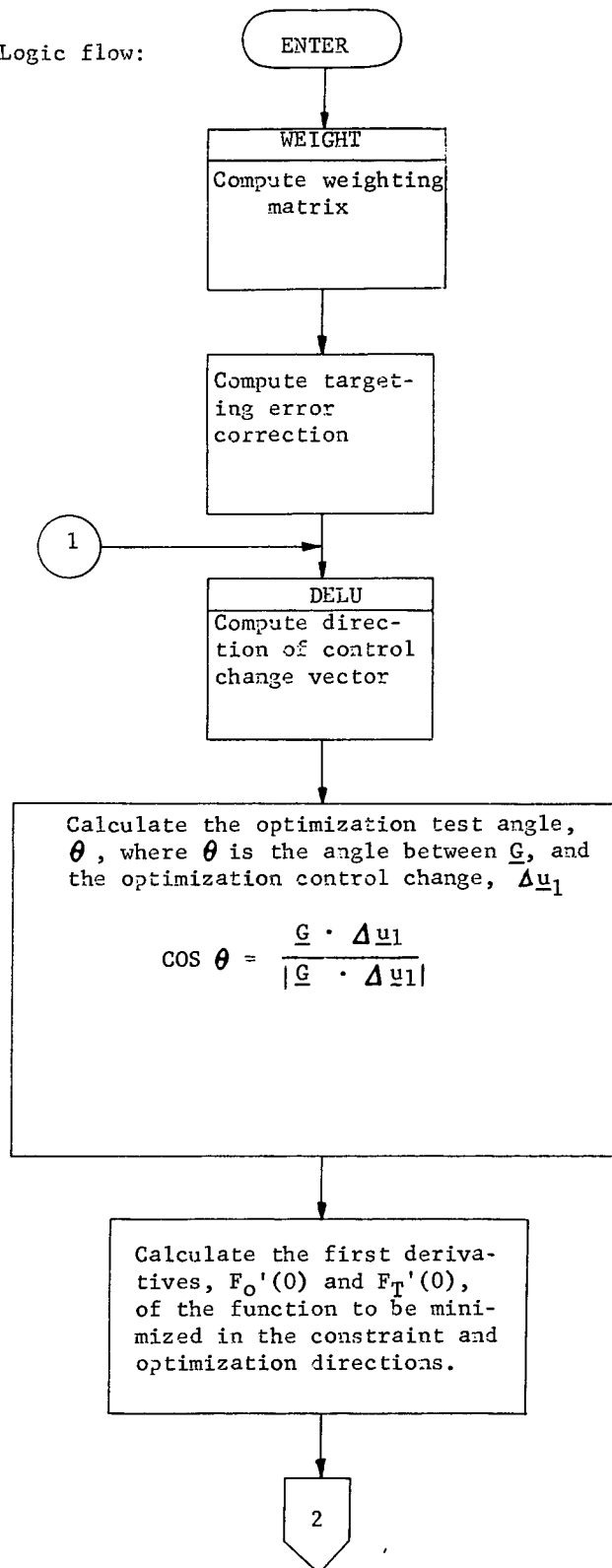
The first derivative evaluated at $\gamma=0$ is then

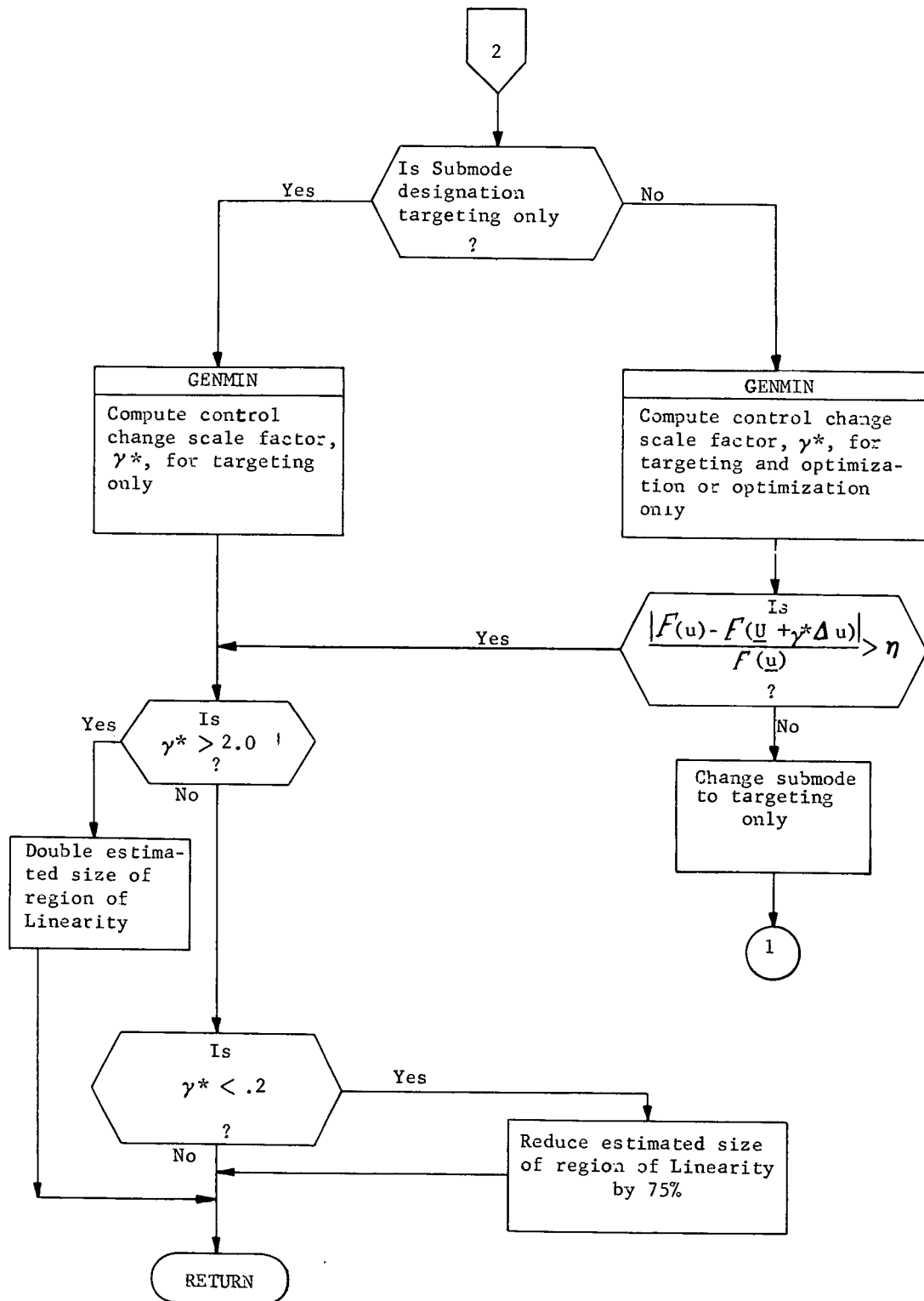
$$F_o'(0) = G^T(U) \Delta U_1$$

Hence

$$F'(0) = \alpha F_T'(0) + \beta F_o'(0)$$

Logic flow:





5.2.9 Subroutine STEP

Purpose: To compute the new control vector

Input:

- o control vector, \underline{U}_{old}
- o control vector scale factor, γ
- o control vector change, $\Delta \underline{U}$
- o dimension of the control vector, M

Output: o new control vector \underline{U}_{new}

Remarks:

The new control step is $\underline{U}_{new} = \underline{U}_{old} + \gamma \Delta \underline{U}$

5.2.10 Subroutine STEST

Purpose: To determine linearly dependent controls among the elements of the control correction vector

Input:

- o sensitivity matrix (the partial derivatives of the target variables with respect to the controls)
- o number of target variables, N
- o number of control variables, M
- o tolerance value, ϵ , determining linear dependency between two controls

Output:

- o number of linearly dependent controls
- o those controls which have been eliminated from the control profile as linearly dependent

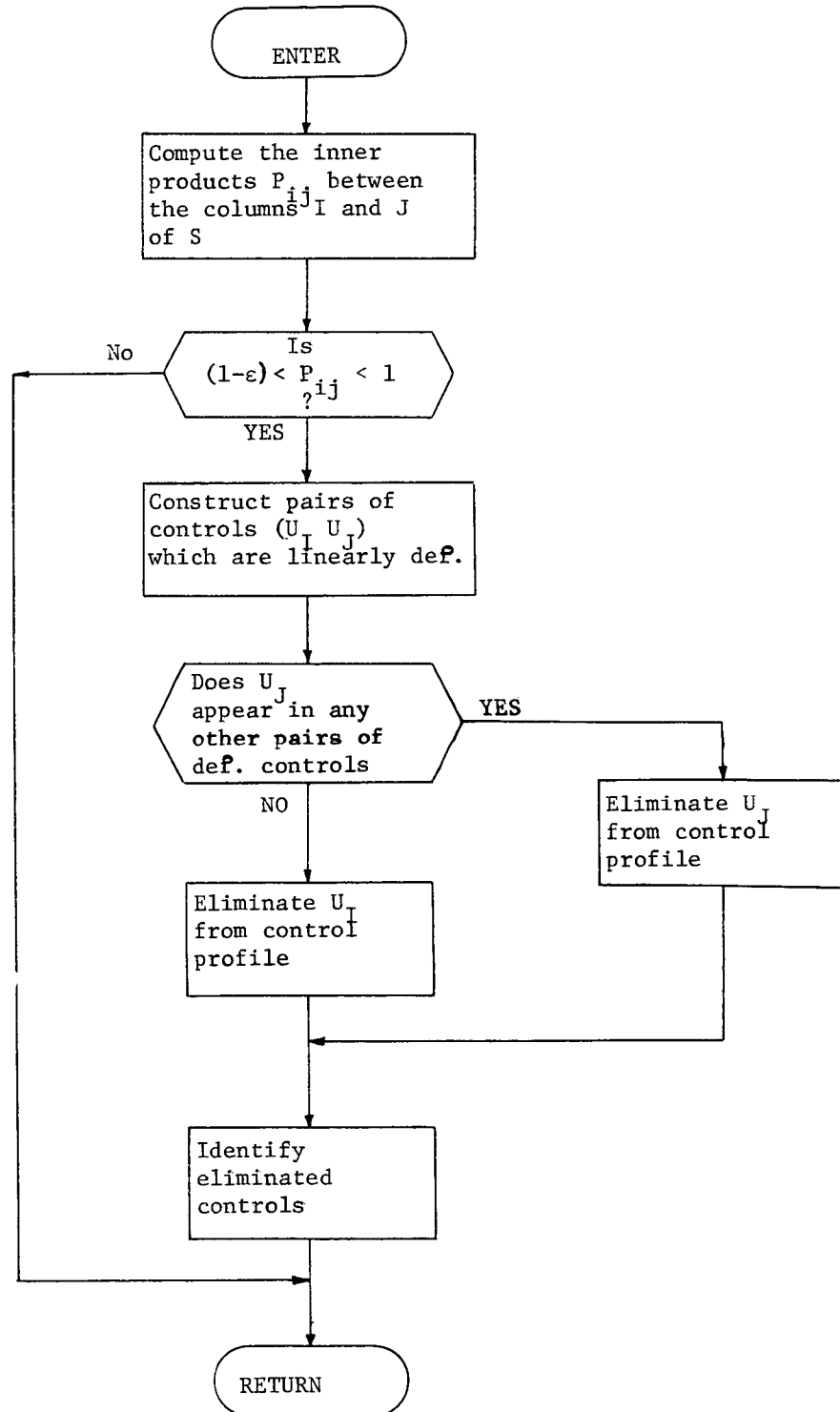
Remarks: The inner products between the columns of the sensitivity matrix, S, are computed where

$$S = \begin{bmatrix} \frac{\partial T_1}{\partial U_1} & \frac{\partial T_1}{\partial U_2} & \cdots & \frac{\partial T_1}{\partial U_m} \\ \frac{\partial T_2}{\partial U_1} & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \frac{\partial T_N}{\partial U_1} & \cdots & \frac{\partial T_N}{\partial T_M} \end{bmatrix}$$

If the value of the inner product, P , between two columns, I and J , is such that

$$(1-\epsilon) \leq P \leq 1$$

then the controls, U_I and U_J , are considered linearly dependent and one of the controls is eliminated from the control profile for at least one iteration (there exists the possibility that within a different region of the control space, P will not satisfy the preceding test condition and the control may again be added to the control profile). For any given pair of linearly dependent controls, the first control is arbitrarily eliminated from the profile unless the second control appears in one or more other linearly dependent pairs. If this situation occurs the second control is eliminated from the profile for at least one iteration.



5.2.11 Subroutine TEST

Purpose: To test for convergence and to make a decision for targeting and/or optimization in the next iteration.

Input:

- o iteration number
- o maximum number of iterations
- o target error index, E
- o limit to which E must be reduced before the targeting submode is discontinued, T_{up}
- o lower bound of E below which the simultaneous targeting and optimization submode is discontinued, T_{low}
- o optimization convergence test angle, θ
- o angle below which the optimization is considered complete, ϵ

Output:

- o submode flag
 - oo target only
 - oo target and optimize
 - oo optimize only
- o convergence flag
 - oo iteration converged
 - oo iteration not converged
 - oo maximum number of iterations reached

Remarks;

The iteration is considered converged and the run is terminated when the performance index is maximized. The test angle θ , which approaches zero as the optimization is completed, is a means of testing the convergence

status (Section 5.2.8 - SIZE).

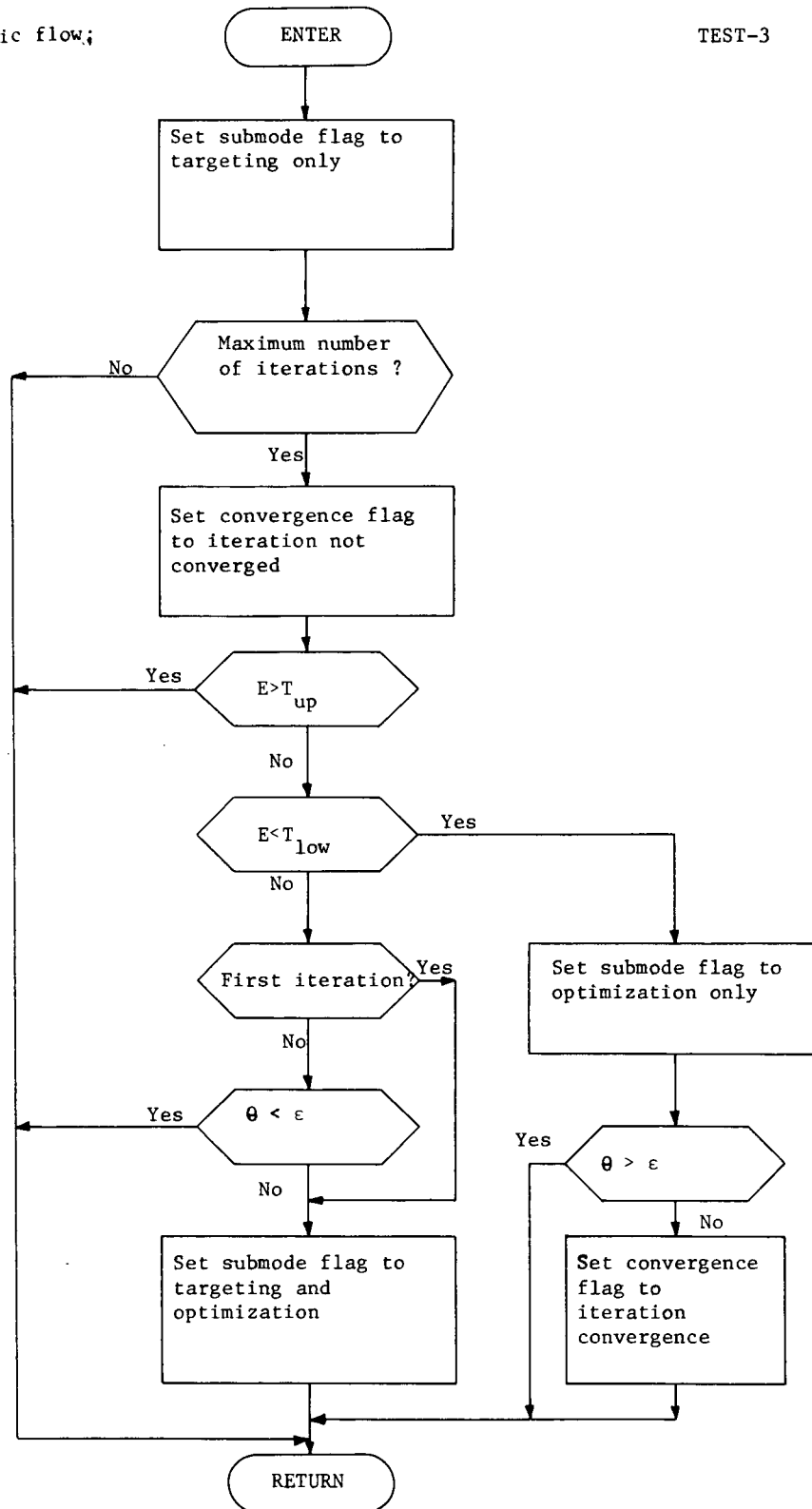
The decision for targeting and/or optimization is based on the current value of the targeting error index, E. If the value of E is greater than T_{up} the targeting submode will be entered. If

$$T_{low} < E < T_{up}$$

then simultaneous targeting and optimization will occur. A value of E less than T_{low} will result in optimization only.

Logic flow;

TEST-3



5.2.12 Subroutine TOM

Purpose: To initialize all parameters and to choose the proper trajectory generation algorithm.

Input: o see the input description of Section 3.1
(Functional Input/Output)

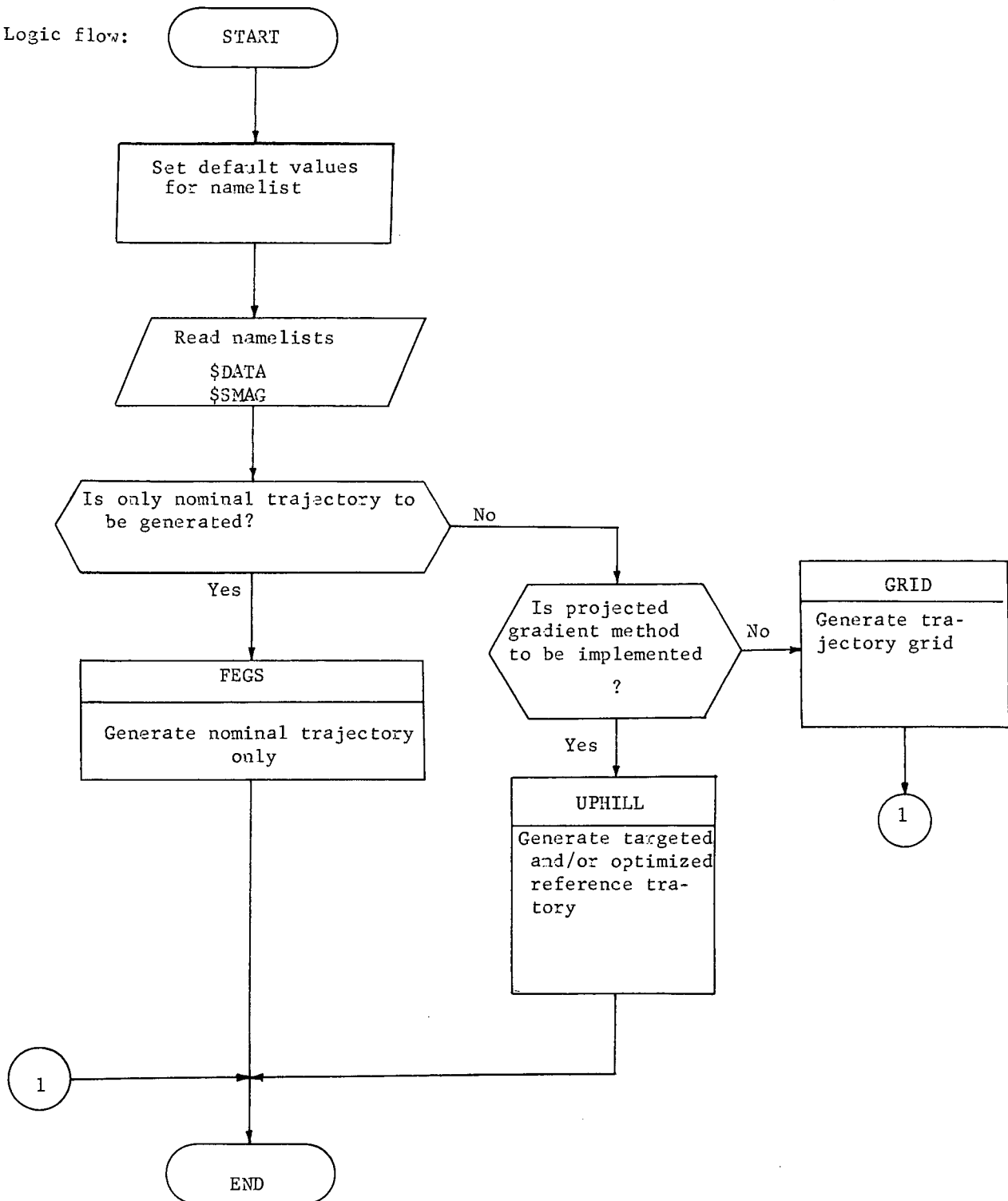
Output: o see output description of Section 3.1
(Functional Input/Output)

Remarks:

TOM initializes parameters through the namelist input \$DATA.

The necessary parameters which are not input assume the default values set in TOM. If the targeting sensitivity matrix and performance gradient for the first iteration are also to be input the namelist \$SMAG will be read.

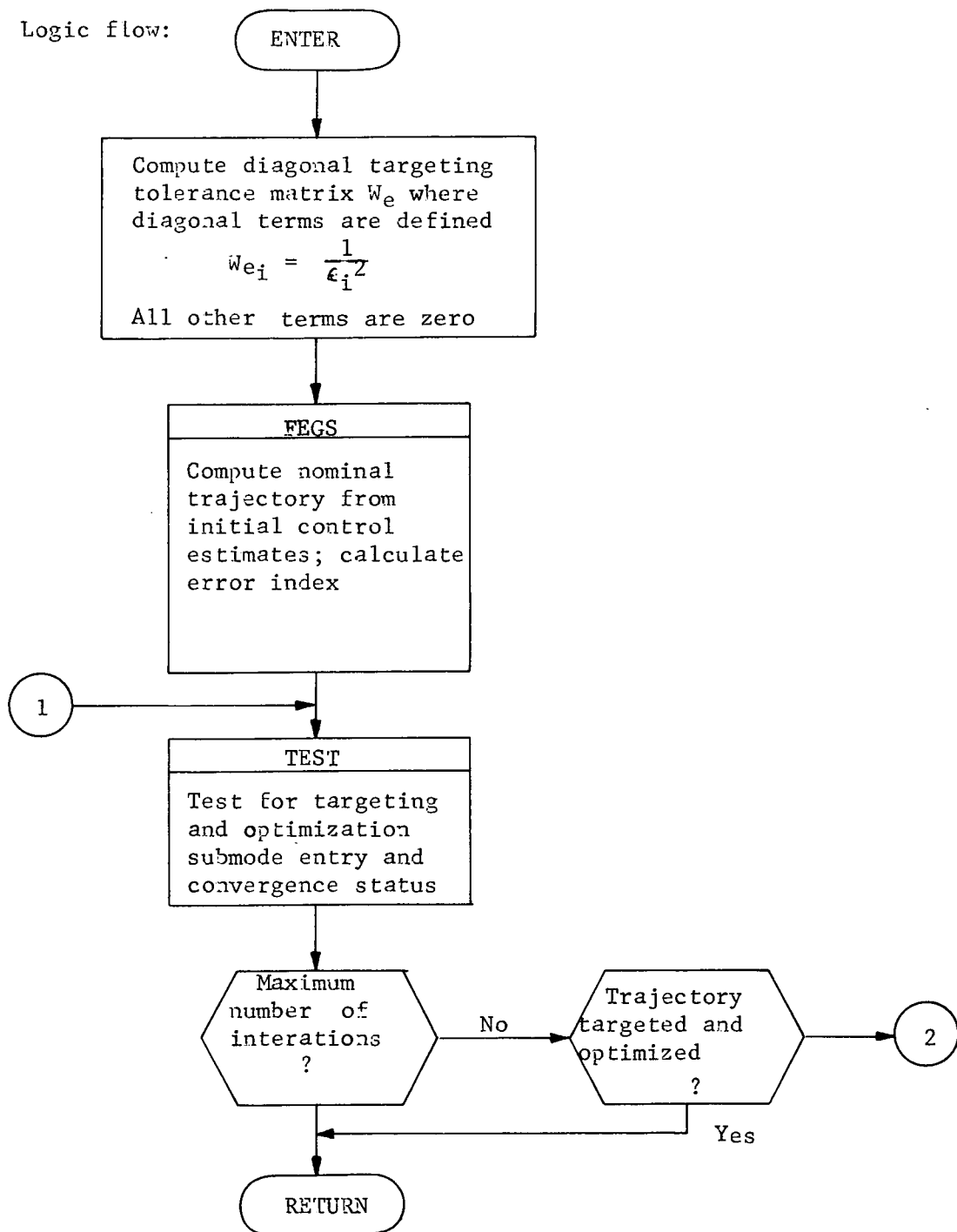
Logic flow:

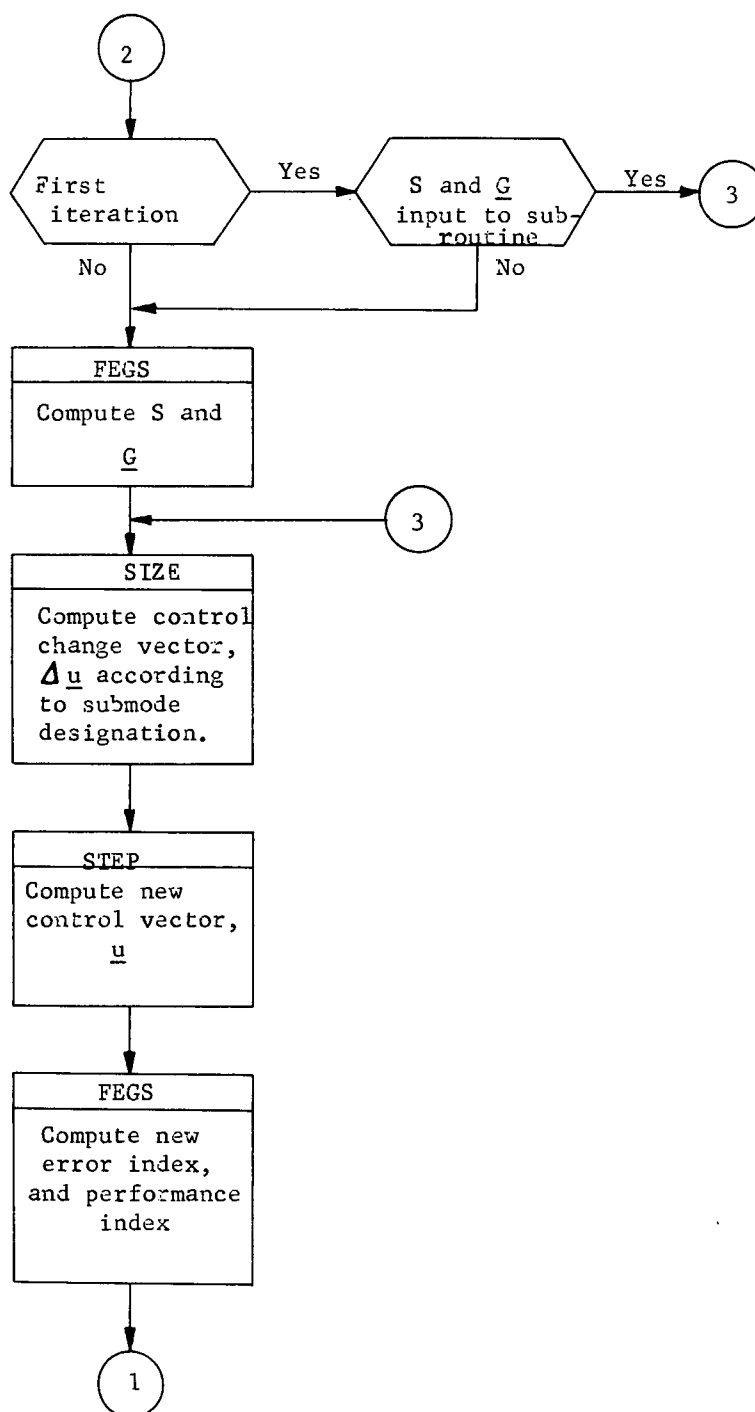


5.2.13. Subroutine UPHILL

- Purpose: To generate a targeted and optimized reference trajectory.
- Input:
- o number of target constraints
 - o number of controls (independent variables)
 - o maximum number of iterations
 - o initial estimate of control vector
 - o error tolerances for targets, ϵ
 - o upper RMS constraint error tolerance
 - o lower RMS constraint error tolerance
 - o estimated radius of region of linearity
 - o percentage of error to be corrected on first iteration
 - o sensitivity matrix S and performance gradient \underline{G} for first iteration
- Output:
- o iteration number
 - o value of performance index
 - o value of the error index, E
 - o optimal control vector, \underline{U}
 - o control change scale factor, γ^*
 - o total control correction, $\Delta \underline{U}$
 - o optimization control correction, $\Delta \underline{U}_1$
 - o constraint control correction, $\Delta \underline{U}_2$
 - o target sensitivities, S
 - o performance gradient, \underline{G}

Logic flow:





5.2.14 Subroutine WEIGHT

Purpose: To generate a weighting matrix

Input:

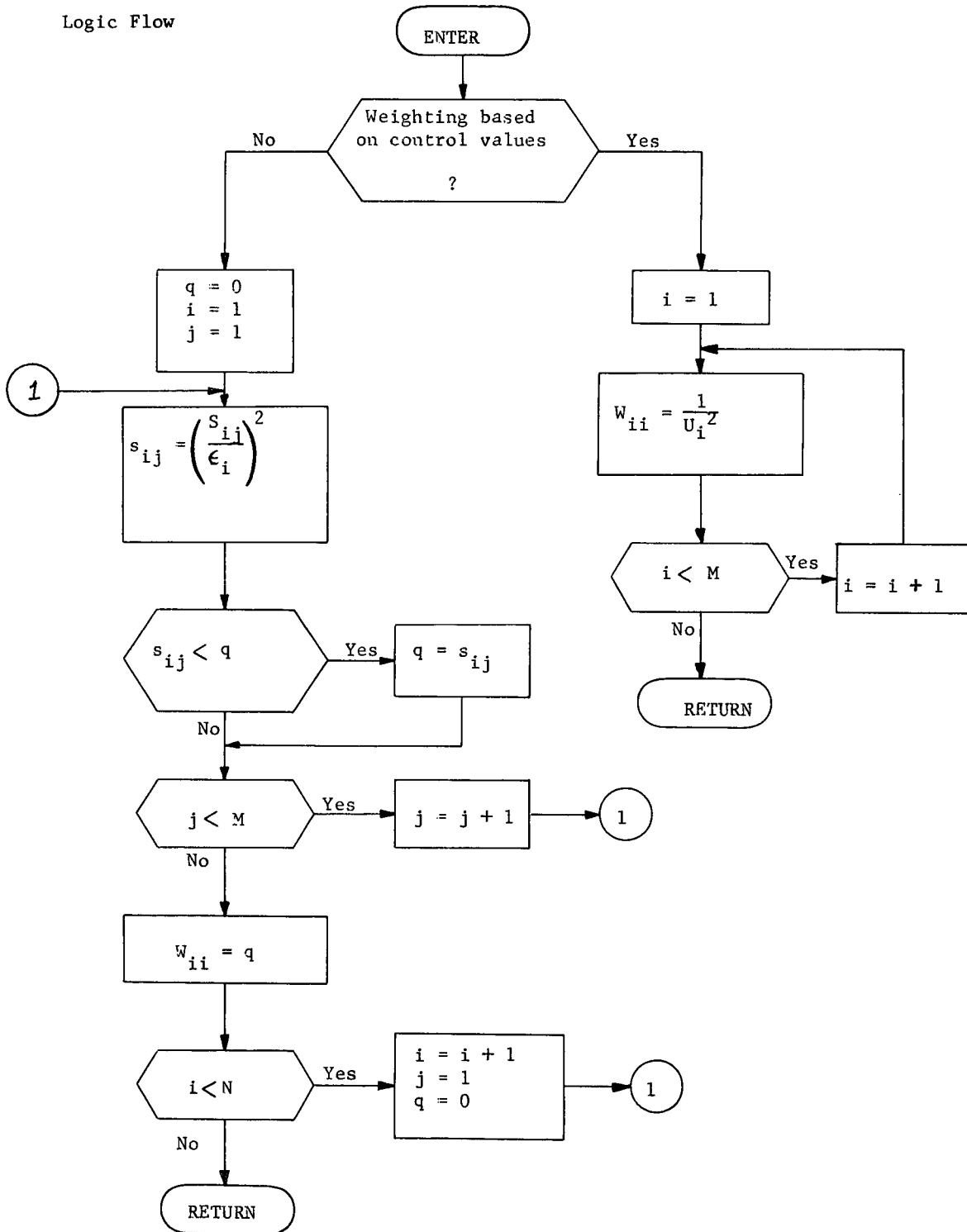
- o nominal control values, U
- o number of controls, M
- o targeting sensitivity matrix, S
- o desired target tolerances, ϵ
- o number of targets, N
- o flag designating type of weighting
 - oo control weighting
 - oo sensitivity weighting

Output: o diagonal weighting matrix, W

Remarks:

The weighting matrix is used in the projected gradient algorithm to emphasize other controls. Two options are available: 1) W based on the largest modulus of the sensitivity elements of each row of the S matrix (see Section 5.2.10, STEST, for a description of S), and 2) W based on the square of the control values. W is a diagonal matrix with all off-diagonal terms equal to zero.

Logic Flow



5.3 Error Analysis Mode

5.3.1 Subroutine CØVP

Purpose: To propagate a covariance matrix from one event time to another

Input:

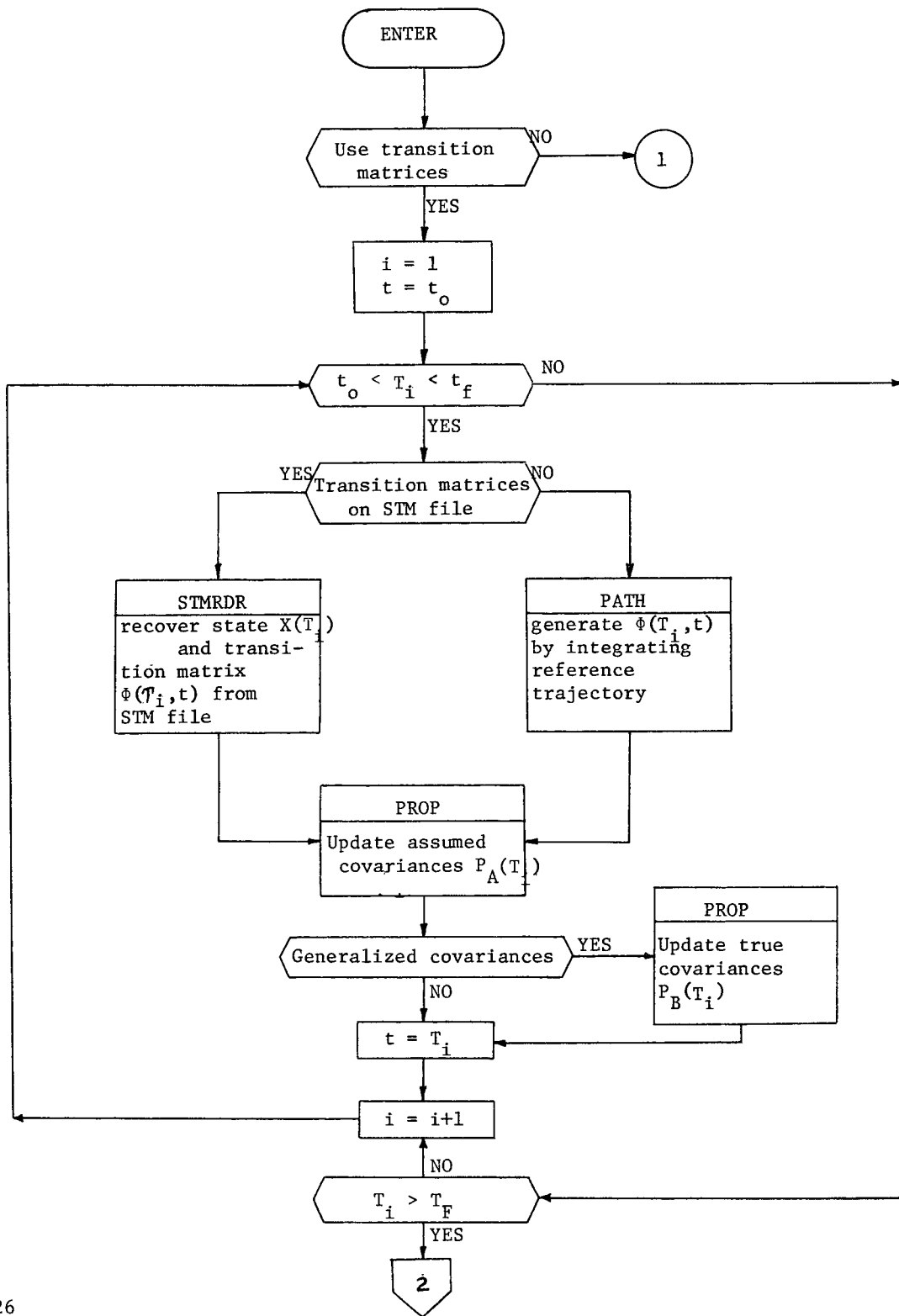
- o initial epoch, t_0
- o initial covariances, P_A and P_B
- o thrust/coast switching times, T_i , $i=1, \dots, N$
- o final epoch, t_F
- o flag for covariance propagation method (state transition matrix or covariance integration)

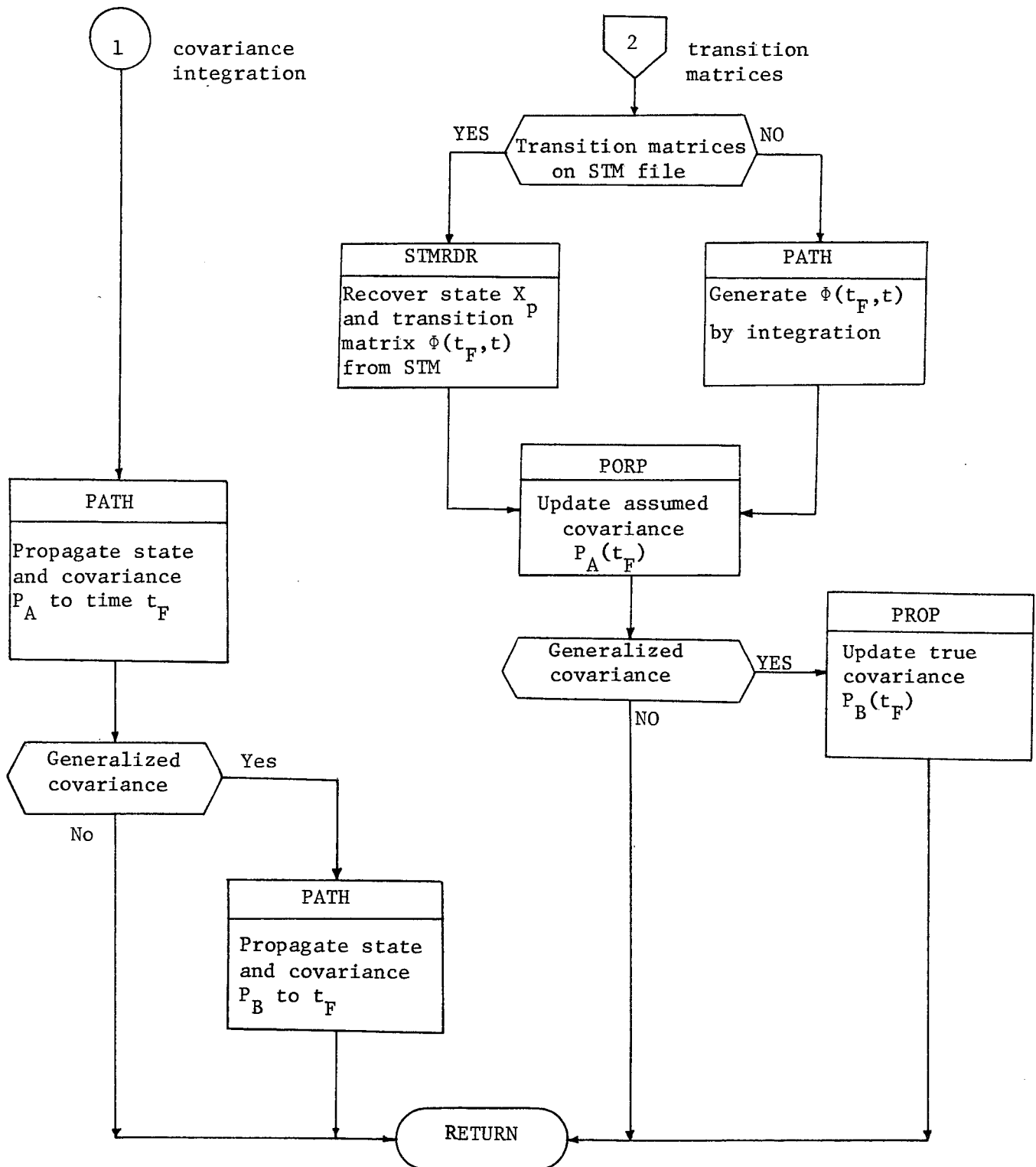
Output:

- o final state, X_F
- o final covariance, P_A and P_B
- o transition matrix, $\phi(t_F, T_j)$
- o dynamic noise, $Q(t_F)$

Remarks:

CØVP will propagate two sets of covariances which are usually the assumed (filter) estimation error covariance and the true (real-world) estimation error covariance. The latter covariance propagation is done only if generalized covariance analysis is desired. Two propagation methods can be chosen: state transition matrices (standard option) and integration of covariance matrix differential equations which are described in more detail in PROP and PATH, respectively.





5.3.2 Program DATA

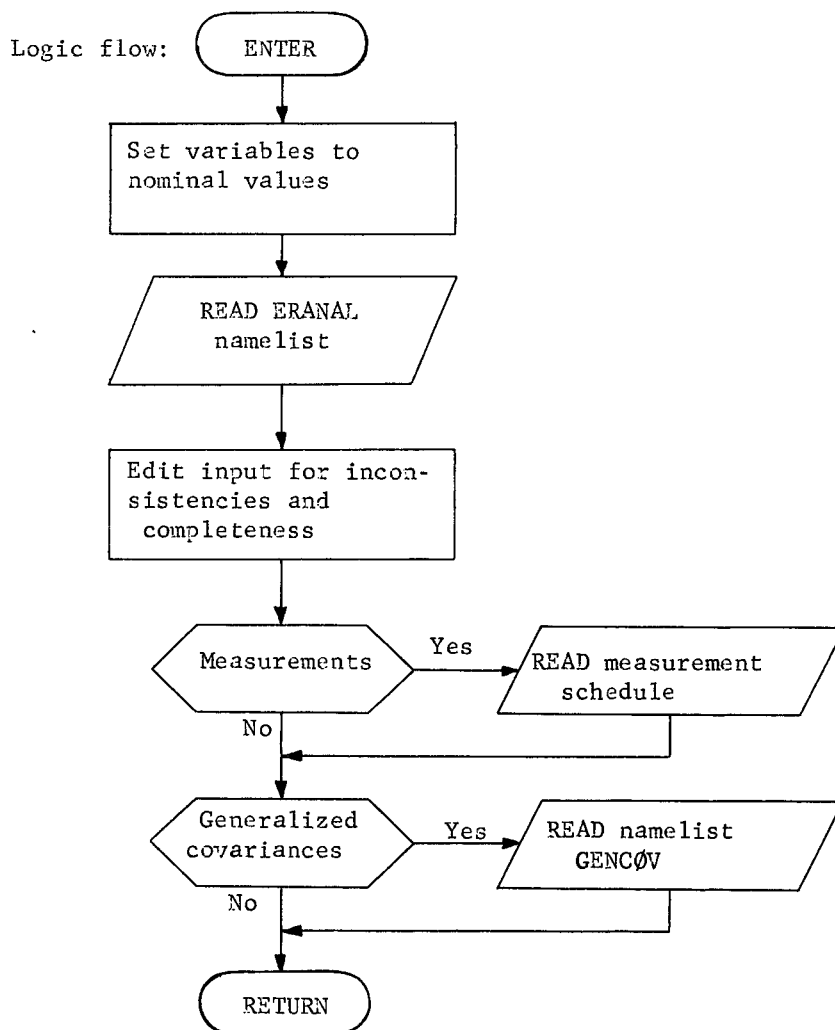
Purpose: To read and edit user input data

Input: (external)

Output: listing and/or error report of user input

Remarks;

DATA reads the first record of STM file to insure that the augmentation parameters to be used in error analysis are a subset of the parameters used at STM generation.



5.3.3 Subroutine DYN0

Purpose: DYN0 computes an effective process noise covariance due to time varying thrust errors

Input: o thrust parameter uncertainties, η
 o correlation time, τ
 o interval length, ΔT
 o basic 6X6 state transition matrix, $\Phi_{k+1,k}$
 o thrust transformation matrix to rotate thrust parameters into cartesian coordinates, h
 o spacecraft state at start and end of interval, (X_k, X_{k+1})

Output: o effective process noise matrix, $\tilde{Q}_{k+1,k}$

Remarks:

The process noise model assumed is a stationary Gauss-Markov process. Since the direct evaluation of a process noise covariance from this model is time consuming, DYN0 computes an analytic approximation to the actual process noise. The justification for this may be found in Appendix 9.3. The equations used are as follows

$$(1) \quad \tilde{Q}_{k+1,k} = R(\Delta t, \tau) \left[\alpha H_{k+1} + \Phi_{k+1,k} H_k \Phi_{k+1,k}^T \right]$$

where

$$(2) \quad \Delta t = t_{k+1} - t_k$$

$$(3) \quad \alpha = \begin{cases} 2, & \Delta t > \tau \\ 0, & \Delta t \leq \tau \end{cases}$$

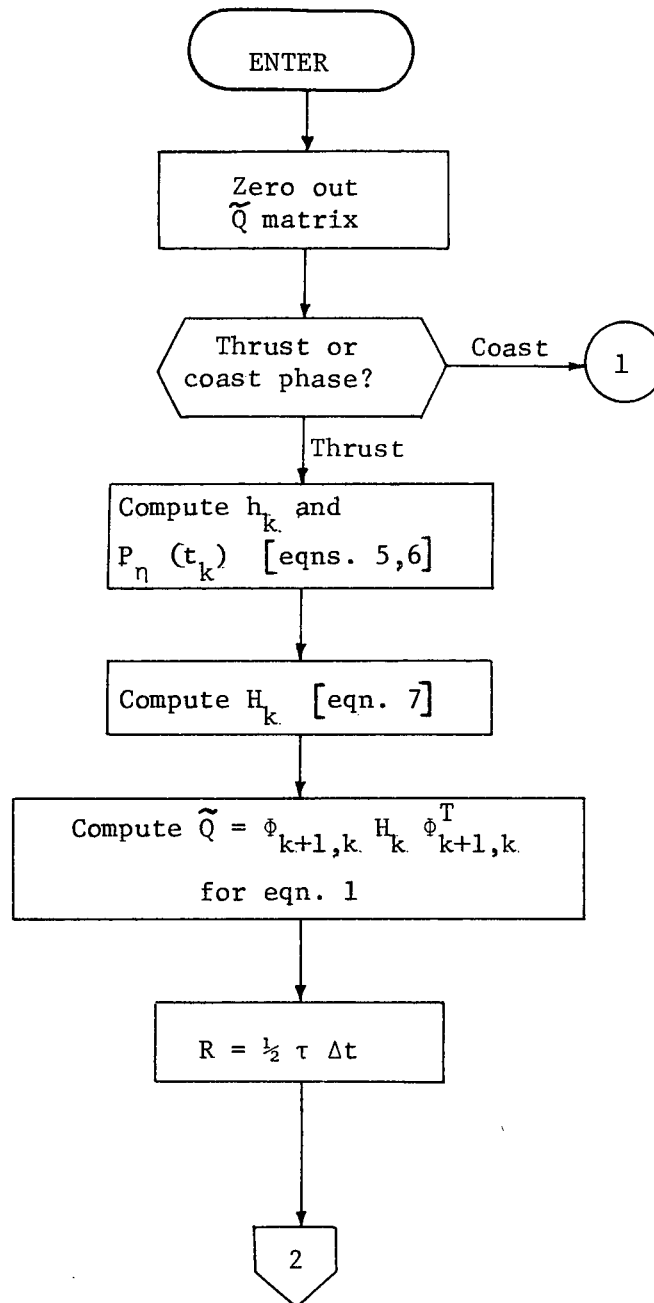
$$(4) \quad R(\Delta t, \tau) = \frac{1}{2} \tau \Delta t$$

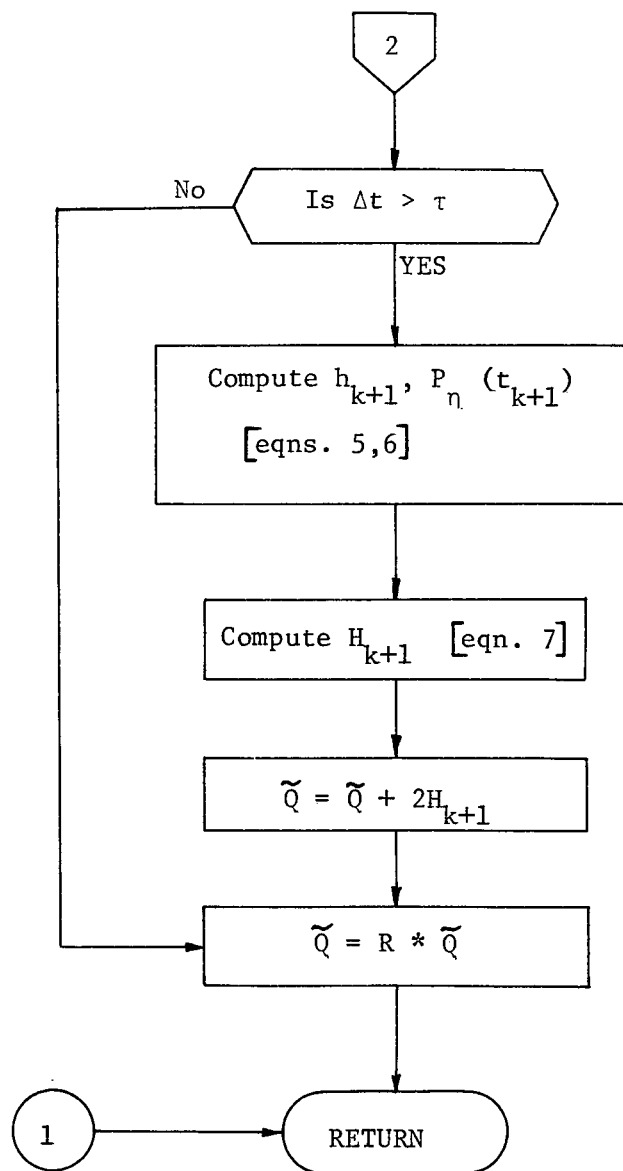
$$(5) \quad h_k = \partial \underline{v}_k / \partial \underline{\eta} (t_k), \quad \underline{v}_k = \text{S/C velocity at } t_k$$

$$(6) \quad P_{\eta} (t_k) = \text{cov} [\eta(t_k)]$$

$$(7) \quad H_k = \begin{bmatrix} 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & h_k P_{\eta} (t_k) h_k^T \end{bmatrix}$$

Since the effective process noise model is invalid over thrusting discontinuities, DYNO assumes that logic exterior to itself has adjusted propagation intervals to guarantee that a thrust on/off event does not occur in the calling interval, (t_k, t_{k+1})





5.3.4 Subroutine FILTER

Purpose: To update augmented state knowledge covariances at a measurement event.

Input:

- o knowledge covariances before event, denoted by superscript $(-)$
- o observation matrices
- o covariance of measurement white noise, R
- o logic control flags
 - oo Kalman-Schmidt, weighted least squares, or user-supplied algorithm
 - oo true or assumed covariance update
- o gain matrix if current update is for true covariances, K

Output:

- o updated knowledge covariances denoted by superscript $(+)$
- o gain matrix

Remarks:

As in subroutine, PROP, all equations below are written for a true covariance update. Wherever differences between true and assumed updates constitute more than simply dropping ignore parameter terms out of an equation, the difference is noted in the logic flow. Timing subscripts are not included here since the entire filtering operation is accomplished at a single time point.

Using the linear measurement model described for TRAKM, Section 5.3.22, results in the following equations for a covariance update. Defining first the measurement residual matrix, J

$$1) \quad J = H_x A + H_s B + H_u D + H_v E + H_w F + R$$

where

$$2) \quad A = P_x^{-T} H_x^T + C_{xs}^{-T} H_s^T + C_{xu}^{-T} H_u^T + C_{xv}^{-T} H_v^T + C_{xw}^{-T} H_w^T$$

$$3) \quad B = P_s^{-T} H_s^T + C_{xs}^{-T} H_x^T + C_{su}^{-T} H_u^T + C_{sv}^{-T} H_v^T + C_{sw}^{-T} H_w^T$$

$$4) \quad D = C_{xu}^{-T} H_x^T + C_{su}^{-T} H_s^T + U_o H_u^T + C_{uv}^{-T} H_v^T + C_{uw}^{-T} H_w^T$$

$$5) \quad E = C_{xv}^{-T} H_x^T + C_{sv}^{-T} H_s^T + C_{uv}^{-T} H_u^T + V_o H_v^T + C_{vw}^{-T} H_w^T$$

$$6) \quad F = C_{xw}^{-T} H_x^T + C_{sw}^{-T} H_s^T + C_{uw}^{-T} H_u^T + C_{vw}^{-T} H_v^T + W_o H_w^T$$

R is the measurement white noise covariance. If the update is to be for assumed covariances, one of the gain matrix subroutines, KSGAIN, WLSGAN, or USRGAN, is called to compute the state and solve-for gain matrices - K_x and K_s . True covariance updates use the gains previously computed by the FILTER pass which updated assumed covariances.

For the Kalman-Schmidt filter, the updates proceed as follows, when K_x and K_s are the state and solve-for parameter gains, respectively.

$$7) \quad P^+ = P^- - K_x A^T$$

$$8) \quad C_{xs}^+ = C_{xs}^- - K_x B^T$$

$$9) \quad C_{xu}^+ = C_{xu}^- - K_x D^T$$

$$10) \quad C_{xv}^+ = C_{xv}^- - K_x E^T$$

$$11) \quad P_s^+ = P_s^- - K_s B^T$$

$$12) \quad C_{su}^+ = C_{su}^- - K_s D^T$$

$$13) \quad C_{sv}^+ = C_{sv}^- - K_s E^T$$

$$14) \quad C_{uv}^+ = C_{uv}^-$$

If, however, the update is for true covariances or assumed covariances for any algorithm other than Kalman-Schmidt, several of the above equations change. While equations 9,10,12,13,14 do not change, equations 7,8, and 11 become, respectively

$$15) \quad P^+ = \left[P^- - K_X A^T \right] - A K_X^T + K_X J K_X^T$$

$$16) \quad C_{XS}^+ = \left[C_{XS}^- - K_X B^T \right] - A K_S^T + K_X J K_S^T$$

$$17) \quad P_S^+ = \left[P_S^- - K_X B^T \right] - B K_S^T + K_S J K_S^T$$

For true covariances the following equations are added

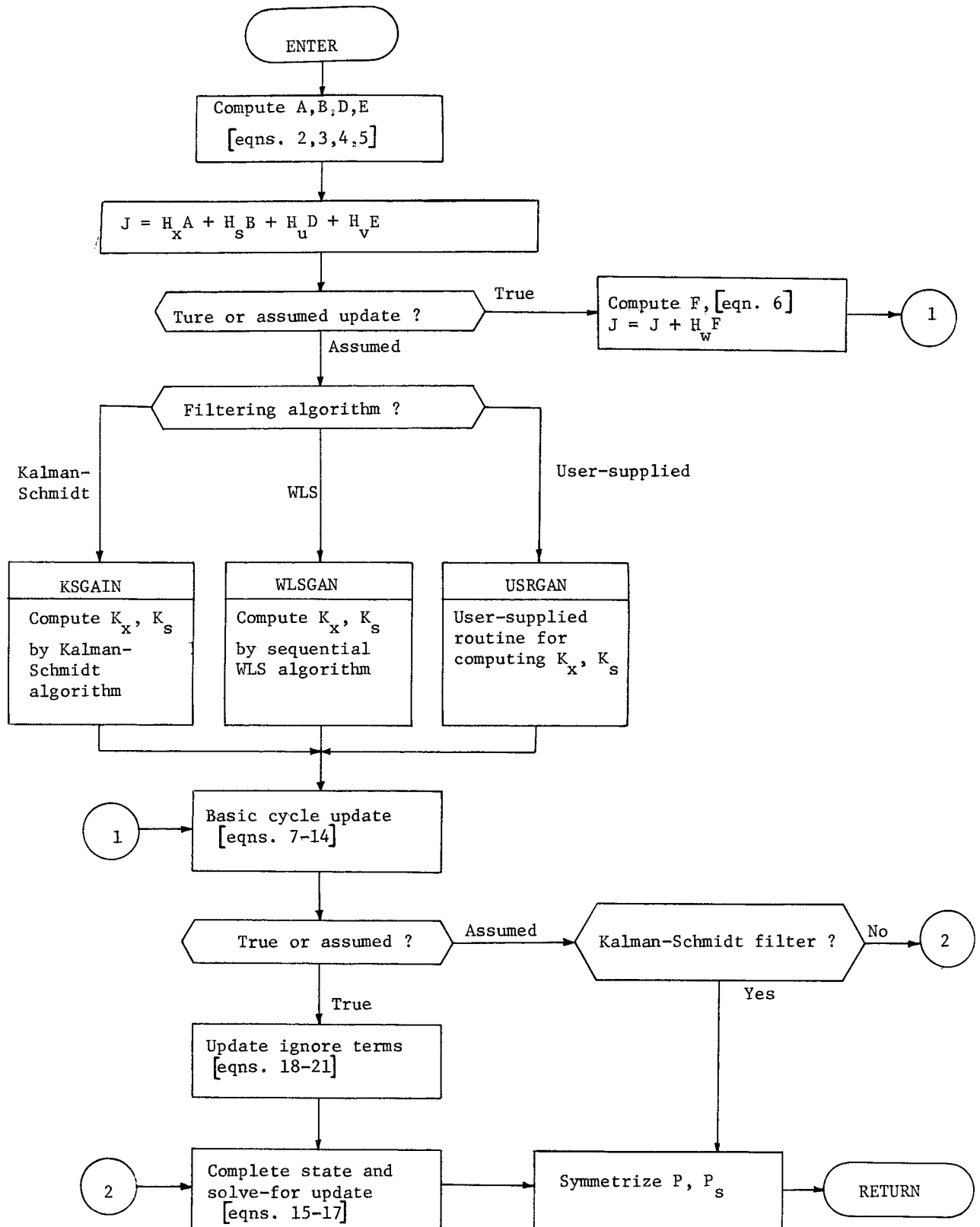
$$18) \quad C_{XW}^+ = C_{XW}^- - K_X F^T$$

$$19) \quad C_{SW}^+ = C_{SW}^- - K_S F^T$$

$$20) \quad C_{uw}^+ = C_{uw}^-$$

$$21) \quad C_{vw}^+ = C_{vw}^-$$

Note that equations 15,16,17 are identical to 7,8,11 with additive terms. Therefore the standard procedure is to execute 7,8,11, and add the necessary terms from 15,16,17 if updating true covariances or using any algorithm other than Kalman-Schmidt.



5.3.5 Subroutine GPRINT

Purpose: To print true covariances and their correlation coefficients,
and dynamic noise covariance.

Input: o true augmented state
 o true dynamic noise covariance

Output: (external)

Remarks:

GPRINT operates on true statistics in a manner analogous to PRINT's
operation on assumed statistics.

5.3.6 Subroutine GUIDM

Purpose: To compute guidance correction requirements and/or update the control error covariance

Input:

- o guidance initiation time, t_I
- o guidance cutoff time, t_c
- o guidance type: impulsive, low thrust, none (update control covariance only)
- o true estimation error covariance, $P_k(t_I)$
- o variation matrix of targets WRT state at t_c , ψ
- o sensitivity matrix of state at t_c WRT thrust controls, S
- o control covariance epoch, t_o
- o control covariance, $P_c(t_o)$
- o transition matrix, $\phi(t_o, t_I)$
- o spacecraft acceleration (a_I and a_c) and mass (m_I and m_c) at t_I and t_c , respectively, and exhaust velocity, c
- o execution errors for impulsive guidance: proportionality (σ_r) and two pointing angles (σ_β and σ_β)

Output:

- o control covariance epoch, t_o
- o control covariance, $P_c(t_o)$

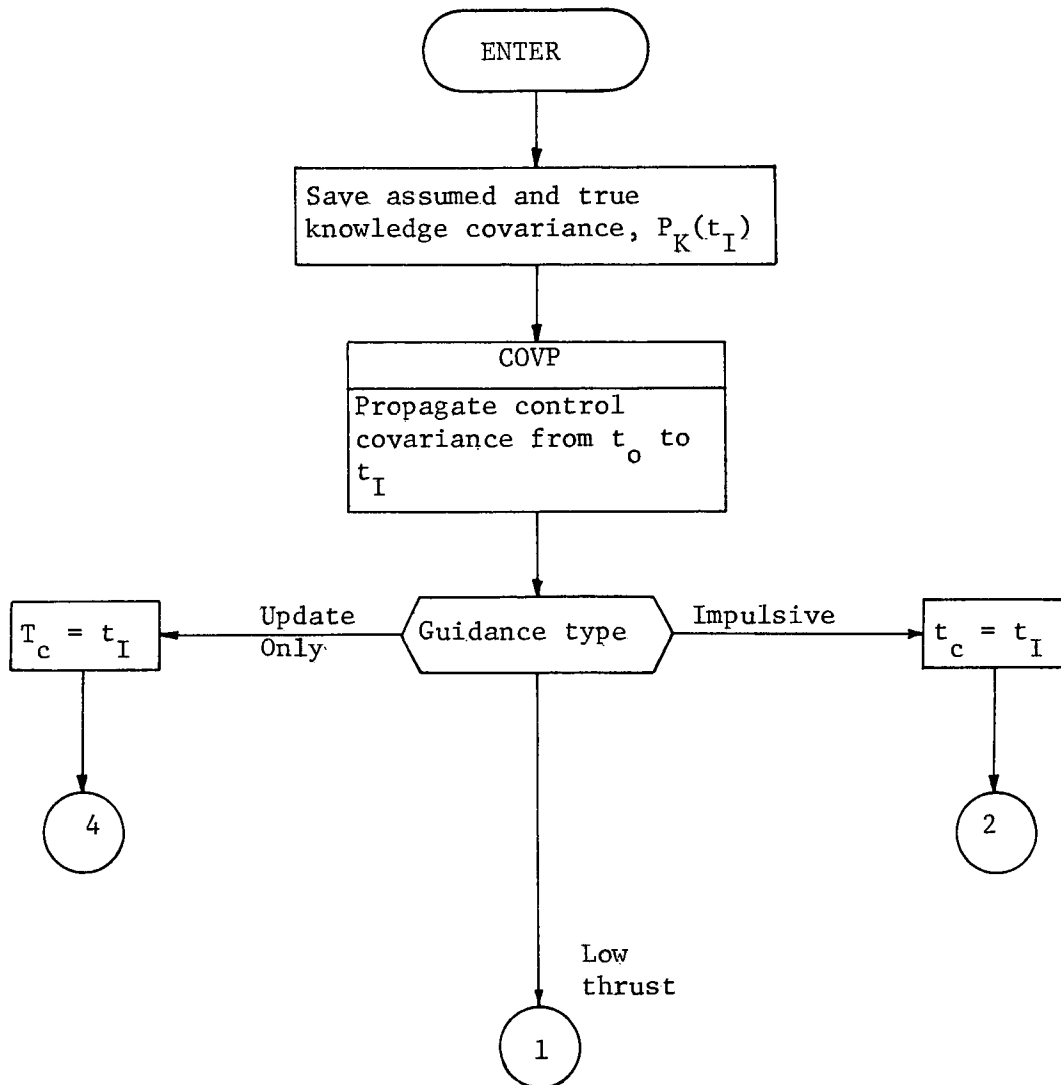
Remarks:

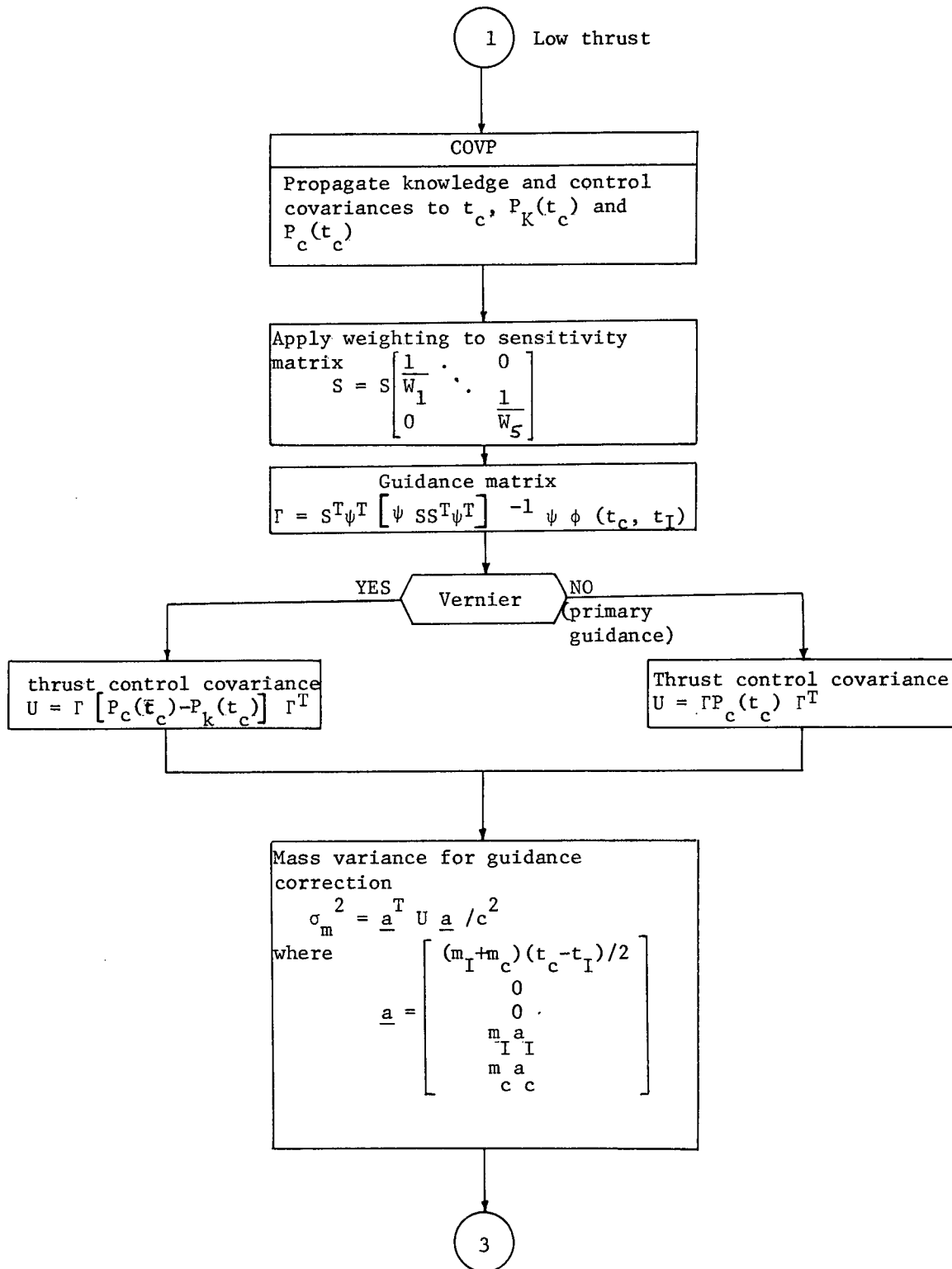
Five thrust controls are allowed: thrust proportionality, two pointing angles, guidance initiation time and guidance termination time. Selective weighting of the controls ($W_1 \dots, W_n$) distributes the control correction accordingly. Whenever the number of controls (either ΔV or low thrust) exceeds the number of targets, the guidance correction algorithm minimizes the weighted control correction. Ensemble

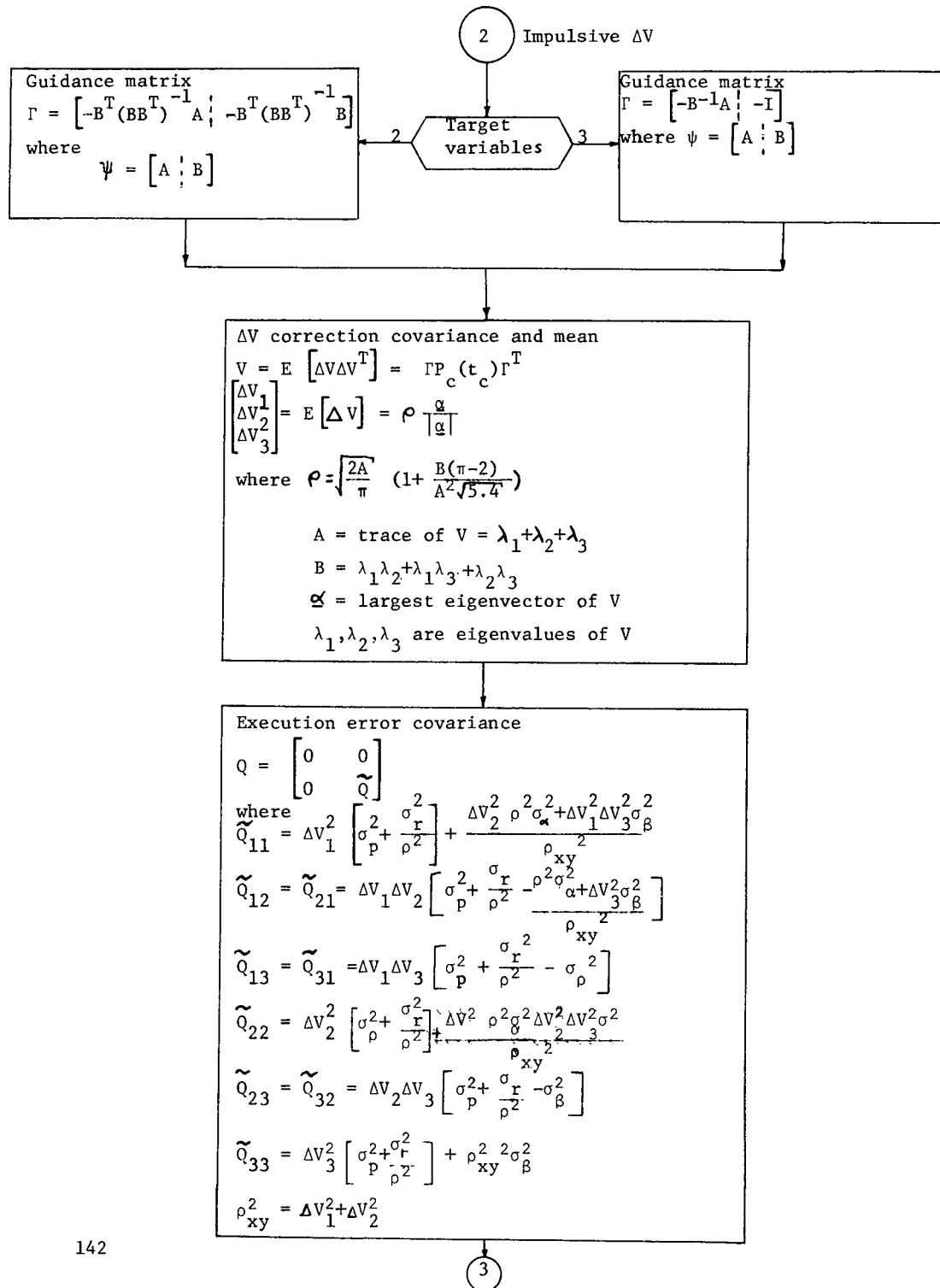
control corrections are pessimistically sized by manner the state control (actual-reference) covariance with the guidance matrix.

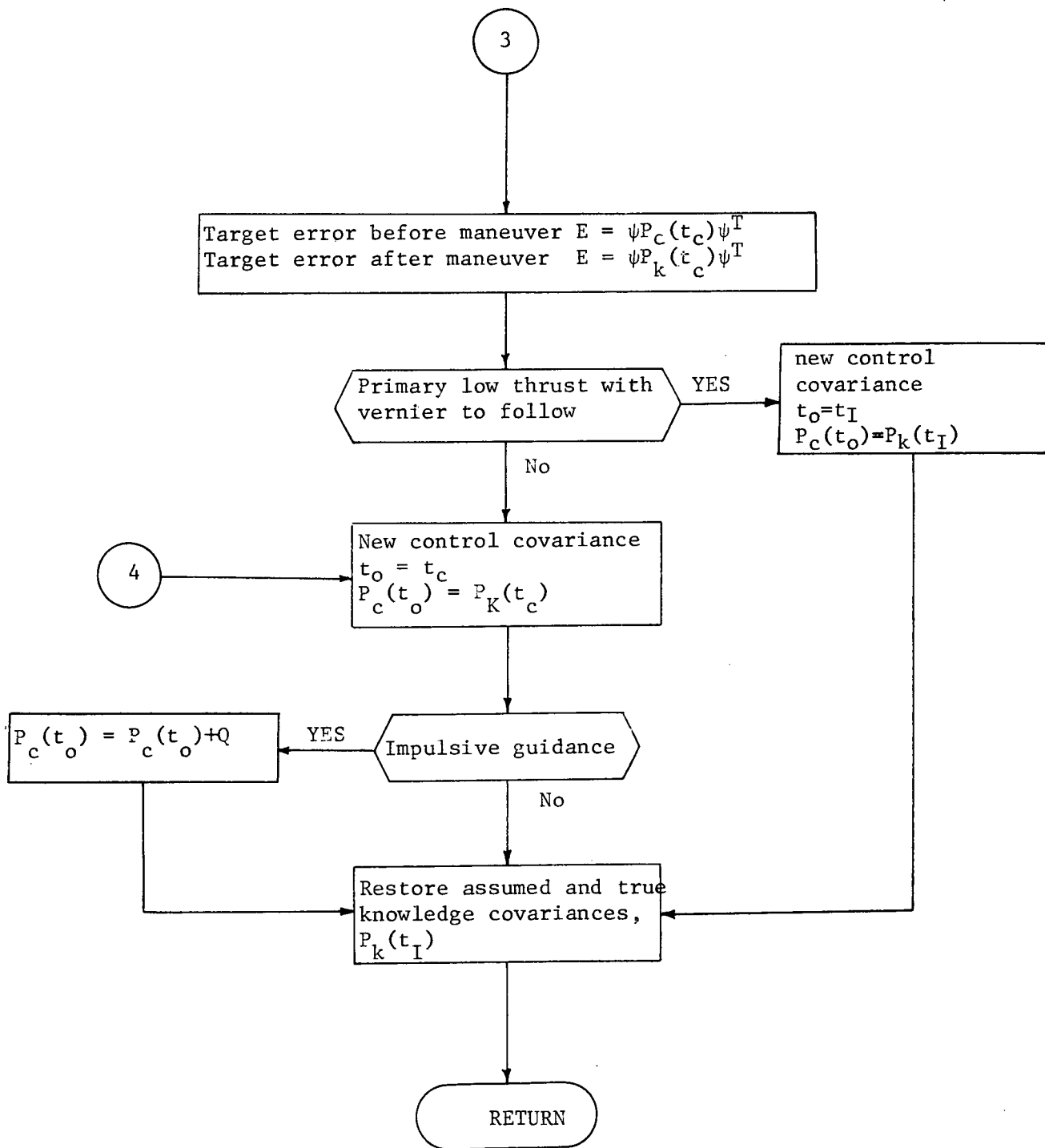
A low thrust "vernier" guidance maneuver is performed between initiation and termination times of a primary guidance maneuver. The vernier removes state error accumulated since initiation of primary guidance or since the last vernier. Whereas the post-maneuver control covariance is normally set equal to the propagated knowledge at guidance termination, for primary guidance with subsequent vernier(s) it is important to set the post-maneuver control covariance equal to the knowledge at guidance initiation.

Impulsive, or ΔV , guidance computes an approximate mean ΔV by the Hoffman-Young formula using the ΔV covariance.









5.3.7 Subroutine KSGAIN

Purpose: To compute gain matrix for Kalman-Schmidt filter.

Input:

- o measurement residual matrix, J
- o cross-covariance of state with measurement residual, A
- o cross-covariance of solve-for parameters with measurement residual, B

Output:

- o gain matrix partitions for state, K_x , and solve-for parameters, K_s

Remarks:

The equations coded are:

$$K_x = A J^{-1}$$

$$K_s = B J^{-1}$$

5.3.8 Program MEAS

Purpose: To control measurement event processing.

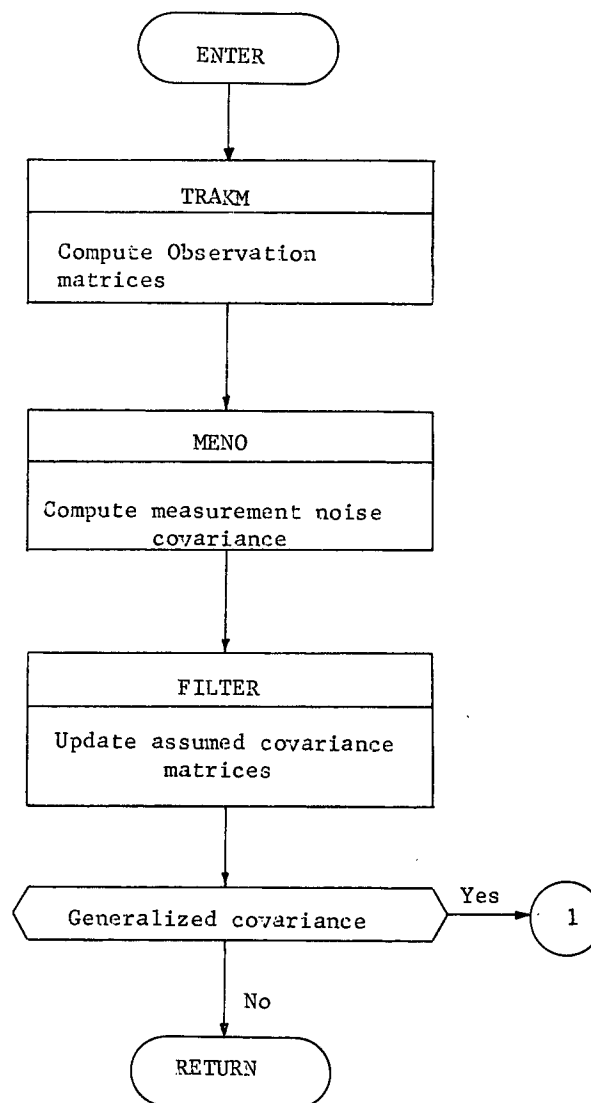
Input:

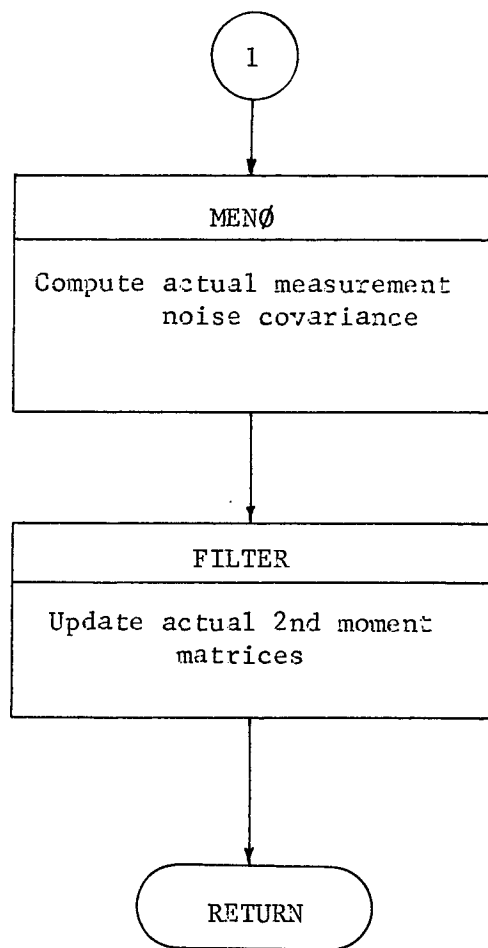
- o current time
- o measurement type
- o generalized covariance flag

Output:

- o updated augmented state covariance matrices

Logic flow:





5.3.9 Subroutine MENO

Purpose: To return the measurement white noise covariance corresponding to the current data type.

Input: o current measurement code
 o array of measurement variances

Output: o measurement white noise covariance, R

Remarks:

According to measurement code, MENO loads the relevant variances from the input array into the current R matrix.

5.3.10 Subroutine PATH

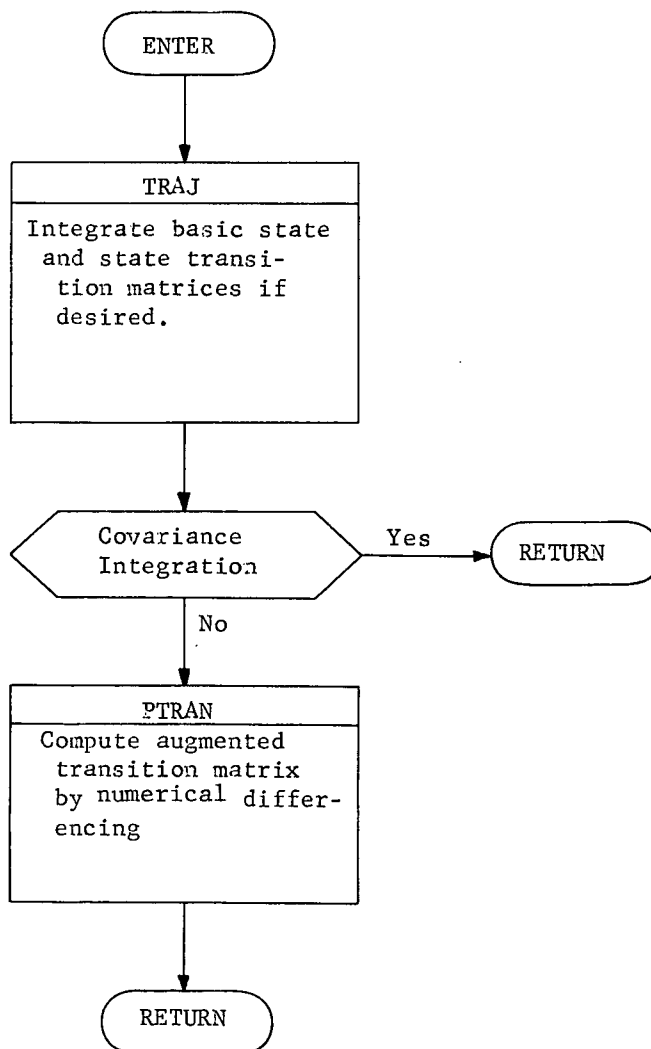
Purpose: To control state propagation and computation of transition matrix

Input:

- current time, t_k
- integration end time, t_{k+1}
- covariance integration flag

Output:

- transition matrix from t_k to t_{k+1}
- state vector at t_{k+1}
- augmented state covariance at t_{k+1}



5.3.11 Subroutine PRED

Purpose: To predict covariance values at some future time.

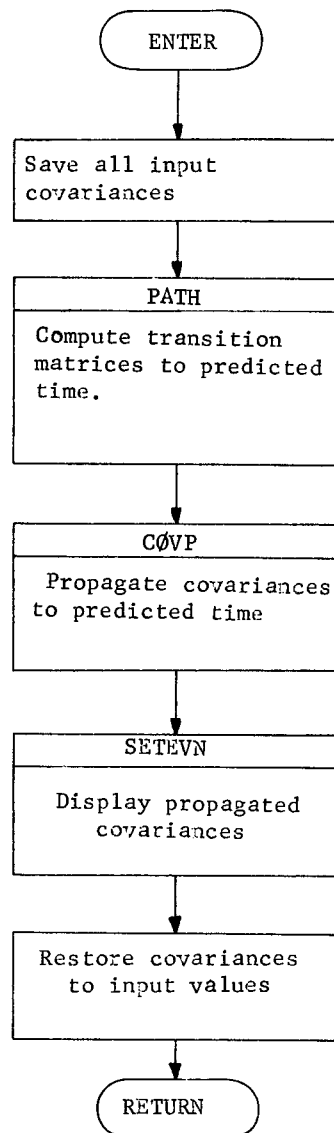
Input:

- time predicted to
- current time
- true and assumed knowledge covariances

Output:

- true and assumed knowledge covariances at predicted time

Logic flow:



5.3.12 Subroutine PRINT

Purpose: To output state vector, covariances and their correlation coefficients, and state transition matrices for assumed statistics

Input:

- o current state
- o current time
- o augmented state covariance matrix
- o state transition matrix
- o assumed dynamic noise covariance

Output: (external)

Remarks:

PRINT transforms data into user-oriented output, computes correlation coefficients, and writes results on an output file.

5.3.13 SUBROUTINE PROP

Purpose: To map covariance matrices at time t_k to time t_{k+1} using the state transition matrix method.

Input:

- o covariances at t_k after all event calculations at that time, denoted by subscript k and superscript (+)
- o state transition matrix partitions over current time interval, denoted by subscript $(k+1,k)$
- o thrust parameter uncertainty
- o flag indicating propagation of true or assumed covariances

Output:

- o Covariances at t_{k+1} before events, denoted by subscript $k+1$ and superscript (-)

Remarks:

Propagation of the augmented state covariance proceeds as

$$(1) \quad \mathcal{P}_{k+1}^- = \phi_{k+1,k} \mathcal{P}_k^+ \phi_{k+1,k}^T + Q_{k+1,k}$$

where

$$(2) \quad \mathcal{P} = \begin{bmatrix} P & C_{xs} & C_{xu} & C_{xv} & C_{xw} \\ C_{xs}^T & P_s & C_{su} & C_{sv} & C_{sw} \\ C_{xu}^T & C_{su}^T & U_o & C_{uv} & C_{uw} \\ C_{xv}^T & C_{sv}^T & C_{uv}^T & V_o & C_{vw} \\ C_{xw}^T & C_{sw}^T & C_{uw}^T & C_{vw}^T & W_o \end{bmatrix}$$

$$(3) \quad \phi = \begin{bmatrix} \Phi & \Theta_{xs} & \Theta_{xu} & 0 & \Theta_{xw} \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}$$

$$(4) \quad Q = \begin{bmatrix} \tilde{Q} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Combining equations 1-4 yields the following equations, where transition matrix subscripts are ignored.

$$(5) \quad P_{k+1}^- = \left[\phi P_k^+ + \theta_{xs} C_{xs_k}^{+T} + \theta_{xu} C_{xu_k}^{+T} + \theta_{xw} C_{xw_k}^{+T} \right] \phi^T + C_{xs_{k+1}}^- \theta_{xs}^T + C_{xu_{k+1}}^- \theta_{xu}^T + C_{xw_{k+1}}^- \theta_{xw}^T + \tilde{Q}_{k+1,k}$$

$$(6) \quad C_{xs_{k+1}}^- = \phi C_{xs_k}^+ + \theta_{xs} P_{s_k}^+ + \theta_{xu} C_{su_k}^{+T} + \theta_{xw} C_{sw_k}^{+T}$$

$$(7) \quad C_{xu_{k+1}}^- = \phi C_{xu_k}^+ + \theta_{xs} C_{su_k}^+ + \theta_{xu} U_o + \theta_{xw} C_{uw_k}^{+T}$$

$$(8) \quad C_{xv_{k+1}}^- = \phi C_{xv_k}^+ + \theta_{xs} C_{sv_k}^+ + \theta_{xu} C_{uv_k}^+ + \theta_{xw} C_{vw_k}^{+T}$$

$$(9) \quad C_{xw_{k+1}}^- = \phi C_{xw_k}^+ + \theta_{xs} C_{sw_k}^+ + \theta_{xu} C_{uw_k}^+ + \theta_{xw} W_o$$

$$(10) \quad P_{s_{k+1}}^- = P_{s_k}^+$$

$$(11) \quad C_{su_{k+1}}^- = C_{su_k}^+$$

$$(12) \quad C_{sv_{k+1}}^- = C_{sv_k}^+$$

$$(13) \quad C_{sw_{k+1}}^- = C_{sw_k}^+$$

$$(14) \quad C_{uv_{k+1}}^- = C_{uv_k}^+$$

$$(15) \quad C_{uw_{k+1}}^- = C_{uw_k}^+$$

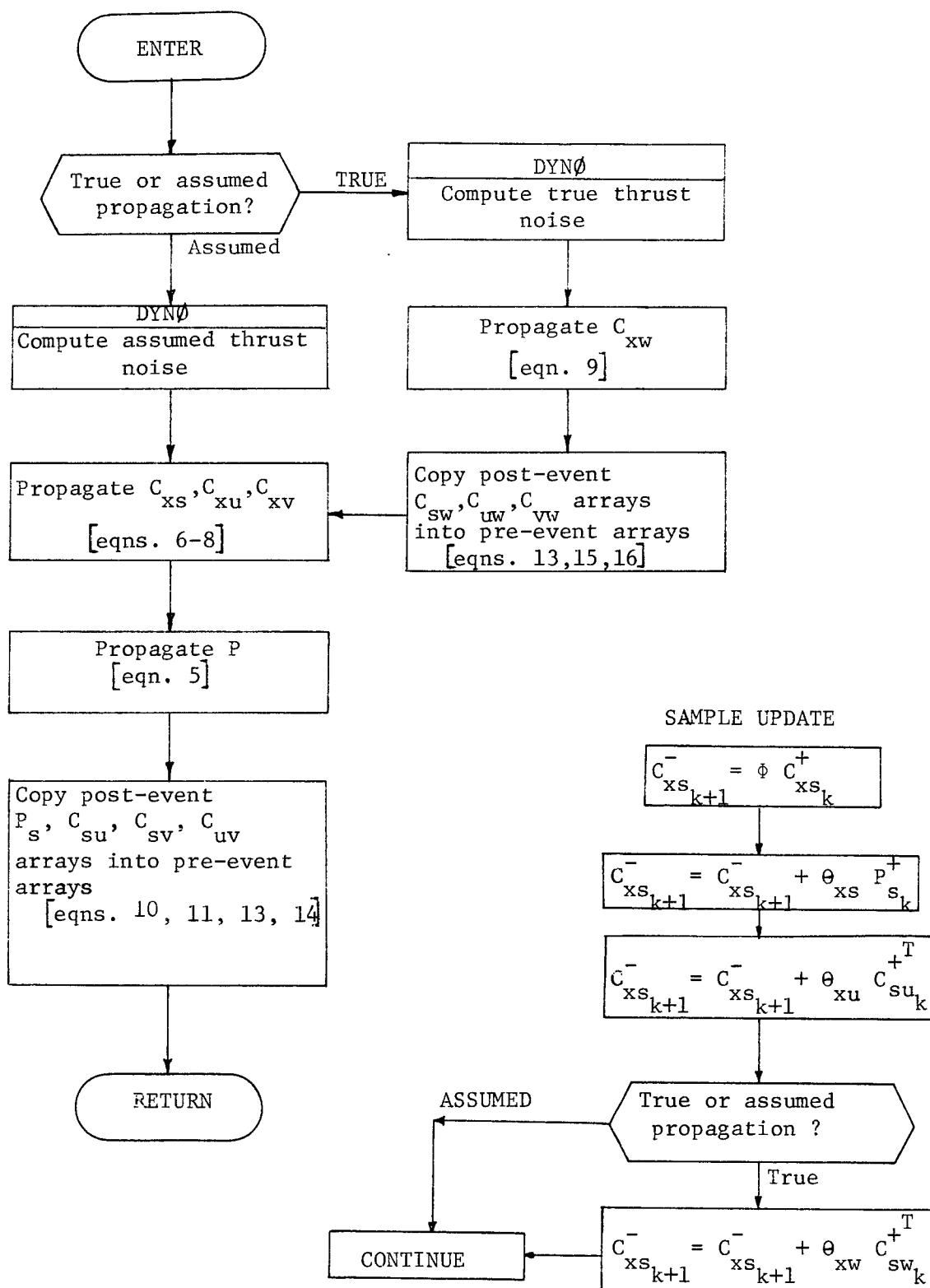
$$(16) \quad C_{vw_{k+1}}^- = C_{vw_k}^+$$

Note that all of the above equations include ignore parameter information, which appears only in true covariance propagation of generalized covariance analysis. The calling sequence to PRØP indicates whether the current propagation is true or assumed covariances. For assumed covariances, all equations and parts of equations deriving from ignore (w) parameters are not processed. The following flow diagram does not show this in detail, so an additional diagram is shown as an example of this ignore parameter by-pass logic.

All matrix multiplications, additions and subtraction are performed by calls to matrix operations routines. In order to avoid programming complexity, calling sequences to these routines are always executed. The logic to prevent unnecessary operations - for example, attempting to compute $C_{xs_{k+1}}^-$, when there are no solve-for parameters - is included within the matrix routines themselves rather than in PRØP.

Logic flow:

PR/P-4



5.3.14 Subroutine PTRAN

Purpose: To generate state transition matrix partitions for dynamic parameters by numerical differencing

Input:

- o spacecraft state at beginning of interval, x_k
- o spacecraft state at end of interval, x_{k+1}
- o interval length, Δt
- o parameter list
- o perturbation magnitude for each parameter

Output: o parameter transition matrix

Remarks:

The augmented state transition matrix, ϕ , may be subdivided as

$$\phi_{k+1,k} = \begin{bmatrix} \phi & \theta_{xs} & \theta_{xu} & 0 & \theta_{xw} \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}_{k+1,k}$$

where the subscript (k+1,k) refers to the time interval (t_k, t_{k+1}) and

$$\theta_{xs_{k+1,k}} = \partial x_{k+1} / \partial s_k, \text{ solve-for}$$

$$\theta_{xu_{k+1,k}} = \partial x_{k+1} / \partial u_k, \text{ dynamic consider}$$

$$\theta_{xw_{k+1,k}} = \partial x_{k+1} / \partial w_k, \text{ ignore.}$$

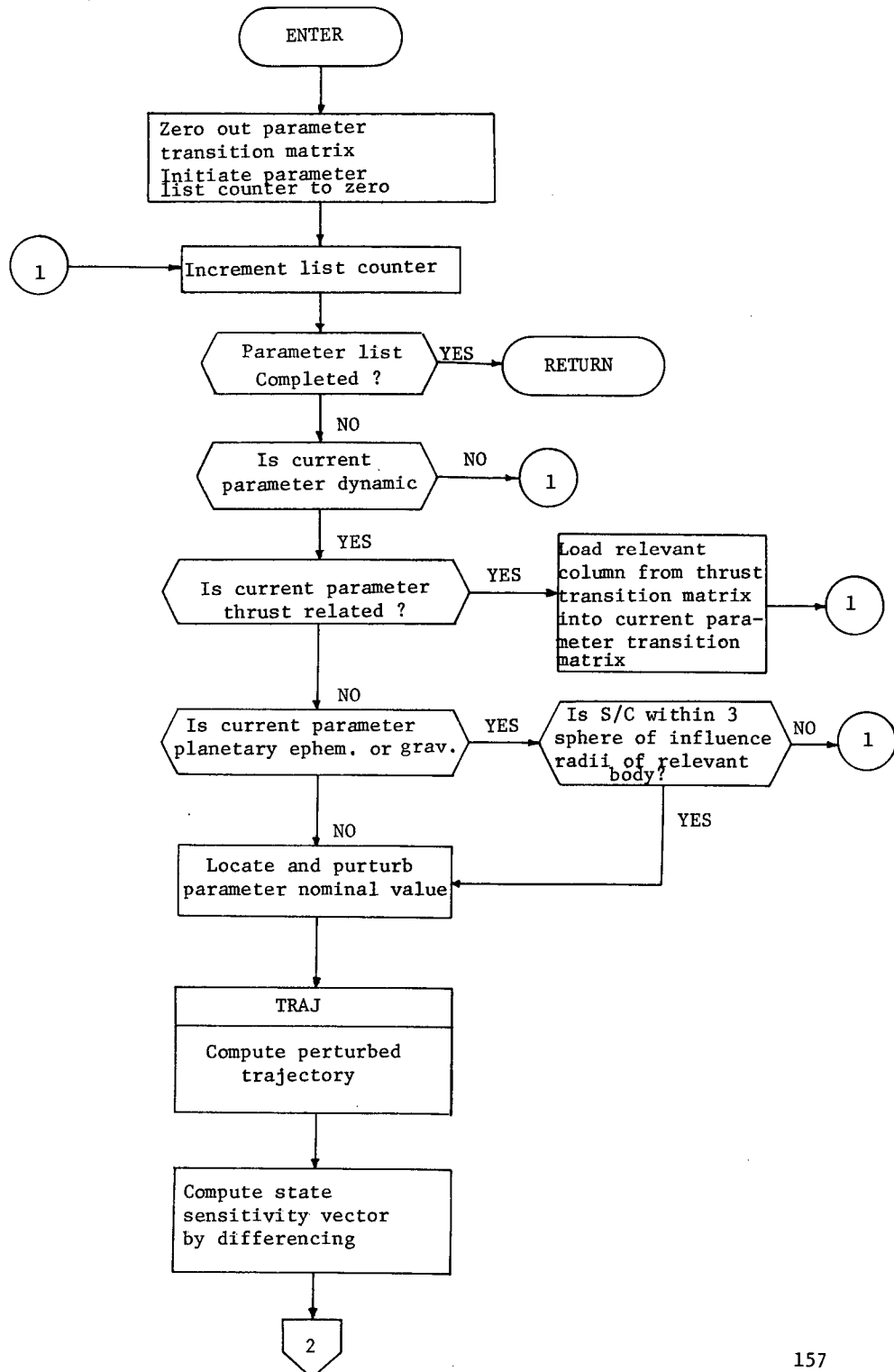
The zero entry in the top row corresponds to the state's independence of measurement consider parameters over time transitions. All sensitivities

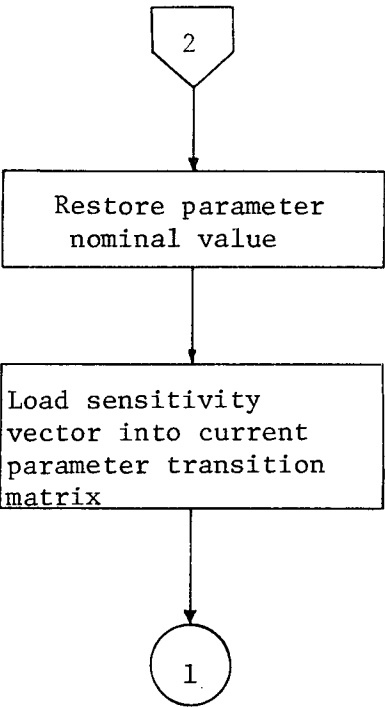
computed in PTRAN are by numerical differencing. All parameter transition matrix elements are computed by PTRAN unless the variational method for thrust is selected by the user.

Note that no mention within the PTRAN flow diagram is ever made of solve-for, consider or ignore status for parameters. This is done for two reasons. First, when the state transition matrix (STM) file is created, no such reference is needed because all parameter sensitivities are generated at once. Parameter type specification is made at error analysis execution time and may change from run to run. Second, if PTRAN is ever used to generate transition matrices in-line with filtering operations, it may be exercised in either of two ways. The first would give PTRAN a parameter list including, in order, the solve-for, consider, and ignore parameters. The transition matrix would be returned and partitioned as necessary for the filtering operations. Or, separate calls to PTRAN could be made for each of the solve-for-consider and ignore options with their distinct parameter list. The extra time necessary for multiple calls is more than saved by eliminating logic necessary to distinguish parameter type within PTRAN.

Logic flow;

PTRAN- 3





5.3.15

Subroutine SCHED

Purpose: To find the next scheduled event

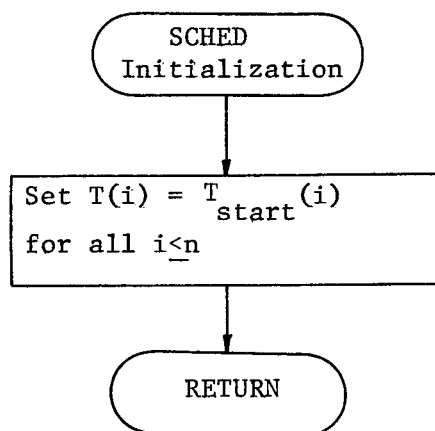
Input:

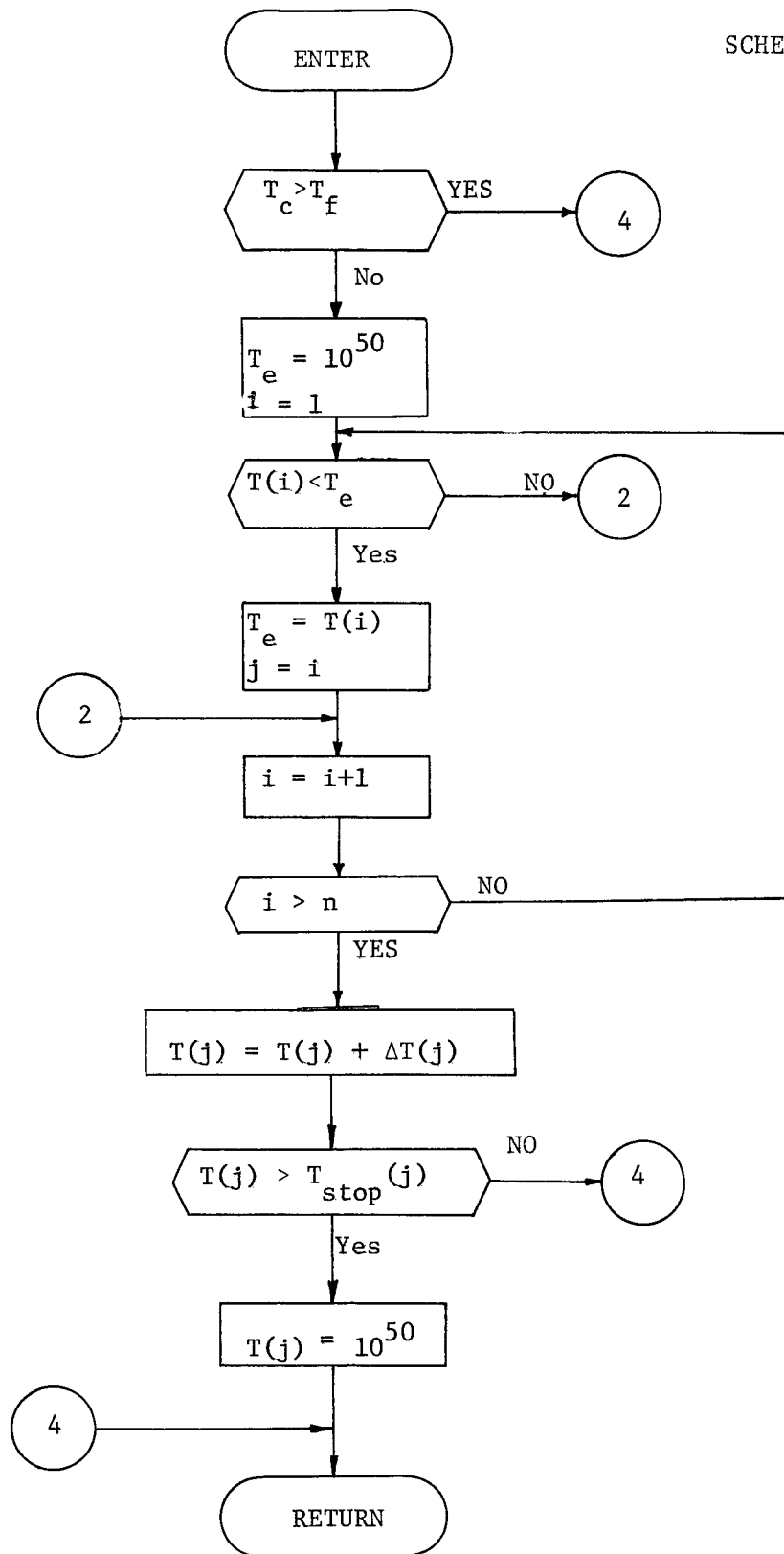
- o Trajectory end time, T_f
- o current trajectory time, T_c
- o number of events, n
- o array of event start times, stop times, time between occurrences of this event, and event code (T_{start} , T_{stop} , ΔT , C)

Output:

- o Time of next event, T_e
- o next event type

Logic flow;





5.3.16 Subroutine SETEVN

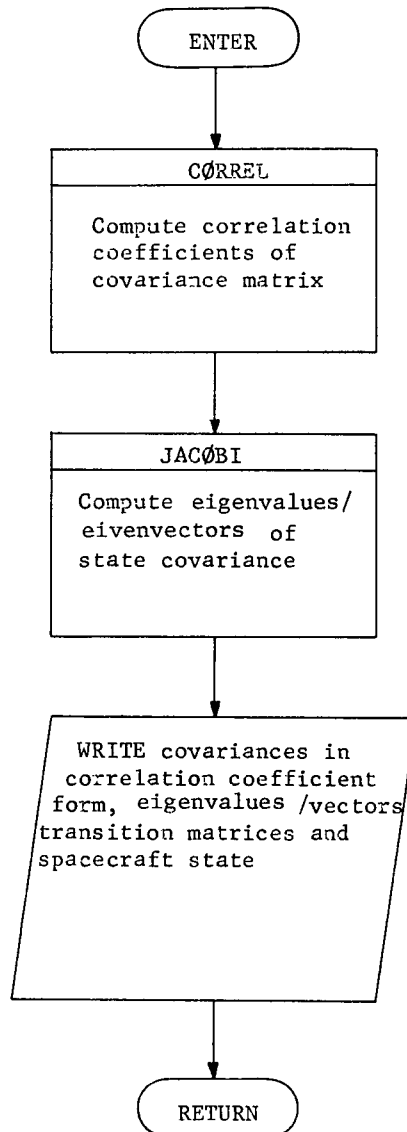
Purpose: To output trajectory information

Input:

- augmented state covariance
- state and parameter transition matrices
- spacecraft state

Output: (external)

Logic flow:



5.3.17 Subroutine STAPRL

Purpose: To compute the negatives of the partials of the spacecraft state WRT station locations

Input:

- o station locations, (R,θ,Ø)
- o Earth obliquity, ε
- o Earth rotation rate, ω
- o universal time from epoch, T
- o current time, t

Output: o partials of state WRT station locations

Remarks:

Let $G = \emptyset + \omega(t-T)$

$$-\frac{\partial X_s}{\partial R} = -\cos\theta \cos G$$

$$-\frac{\partial X_s}{\partial \theta} = R \sin\theta \cos G$$

$$-\frac{\partial X_s}{\partial \emptyset} = R \cos\theta \sin G$$

$$-\frac{\partial Y_s}{\partial R} = (\sin\epsilon \sin\theta + \cos\epsilon \cos\theta \sin G)$$

$$-\frac{\partial Y_s}{\partial \theta} = R \cos\epsilon \sin\theta \sin G - R \sin\epsilon \cos\theta$$

$$-\frac{\partial Y_s}{\partial \emptyset} = -R \cos\epsilon \cos\theta \cos G$$

$$-\frac{\partial Z_s}{\partial R} = \sin\epsilon \cos\theta \sin G - \cos\epsilon \sin\theta$$

$$-\frac{\partial Z_s}{\partial \theta} = - (R \sin\epsilon \sin\theta \sin G + R \cos\epsilon \cos\theta)$$

$$-\frac{\partial Z_s}{\partial \emptyset} = R \sin\epsilon \cos\theta \cos G$$

$$-\frac{\partial \dot{X}_s}{\partial R} = \omega \cos\theta \sin G$$

$$-\frac{\partial \dot{X}_s}{\partial \theta} = -\omega R \sin \theta \cos G$$

$$-\frac{\partial \dot{X}_s}{\partial \phi} = \omega R \cos \theta \cos G$$

$$-\frac{\partial \dot{Y}_s}{\partial R} = -\omega \cos \theta \cos \epsilon \cos G$$

$$-\frac{\partial \dot{Y}_s}{\partial \theta} = \omega R \cos \epsilon \sin \theta \cos G$$

$$-\frac{\partial \dot{Y}_s}{\partial \phi} = \omega R \cos \epsilon \cos \theta \sin G$$

$$-\frac{\partial \dot{Z}_s}{\partial R} = \omega \sin \epsilon \cos \theta \cos G$$

$$-\frac{\partial \dot{Z}_s}{\partial \theta} = -\omega R \sin \epsilon \sin \theta \cos G$$

$$-\frac{\partial \dot{Z}_s}{\partial \phi} = -\omega R \sin \epsilon \cos \theta \sin G$$

5.3.18 Subroutine: STMGEN

Purpose: To create an STM file containing the integrated state and augmented state transition matrix for all events except prediction events.

Input:

- o event schedules
- o final time
- o start time
- o STM file generation data as described in Error Analysis Functional I/O Section 3.2

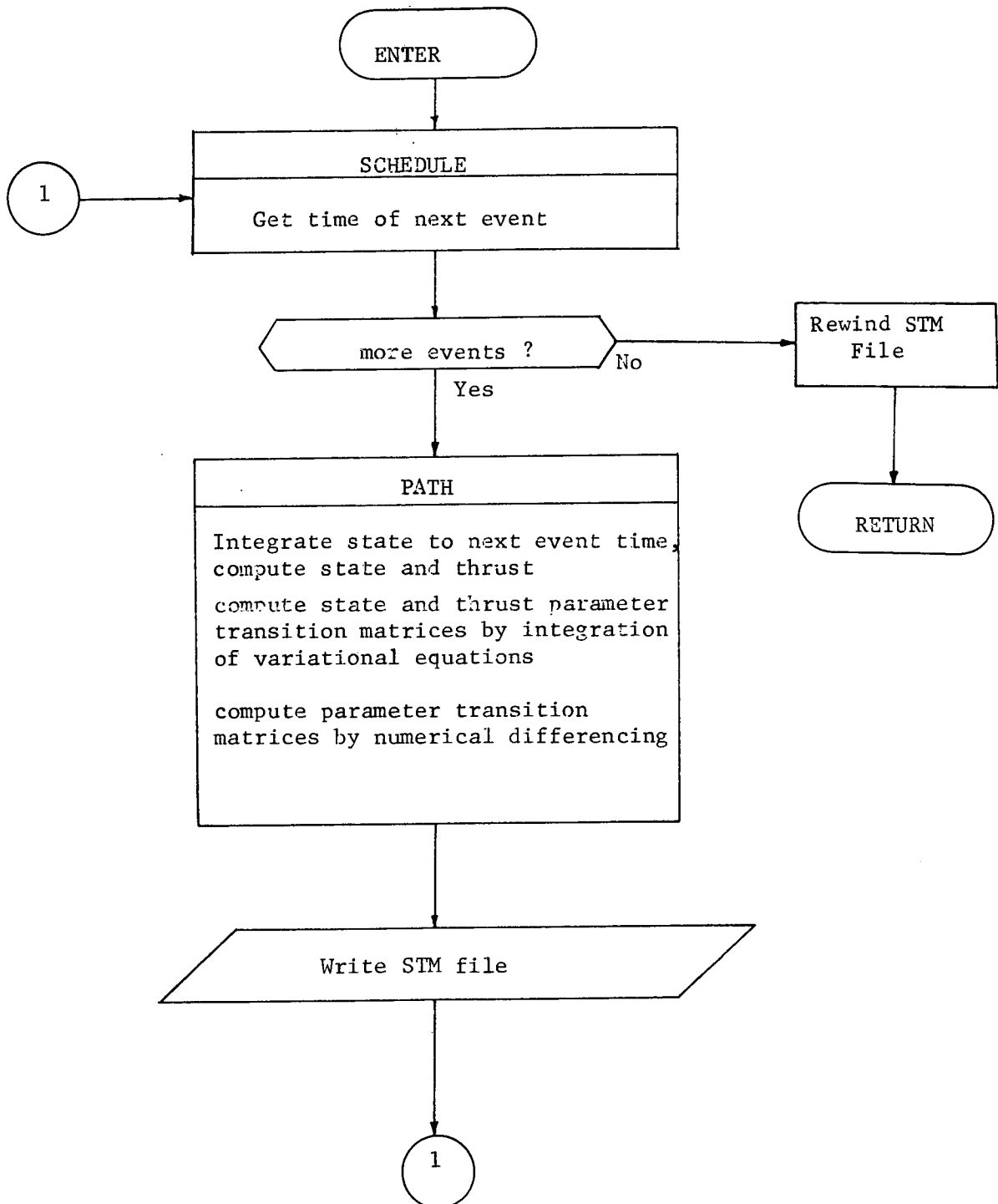
Output: (External)

Remarks:

The layout of the STM file is:

- oo record 0: array of parameter numbers augmented to the state at STM file generation (used by DATA for error checking)
- oo records 1 - N:
 - word 1 - event time
 - 2 - event type
 - 3-8 - spacecraft state vector
 - 9-44 - state transition matrix
 - 45-224 - parameter transition matrix

At STM file generation, no distinction is made between solve-for and consider parameters. A single parameter transition matrix is computed. Parameters will be separated in subroutine STMRDR.



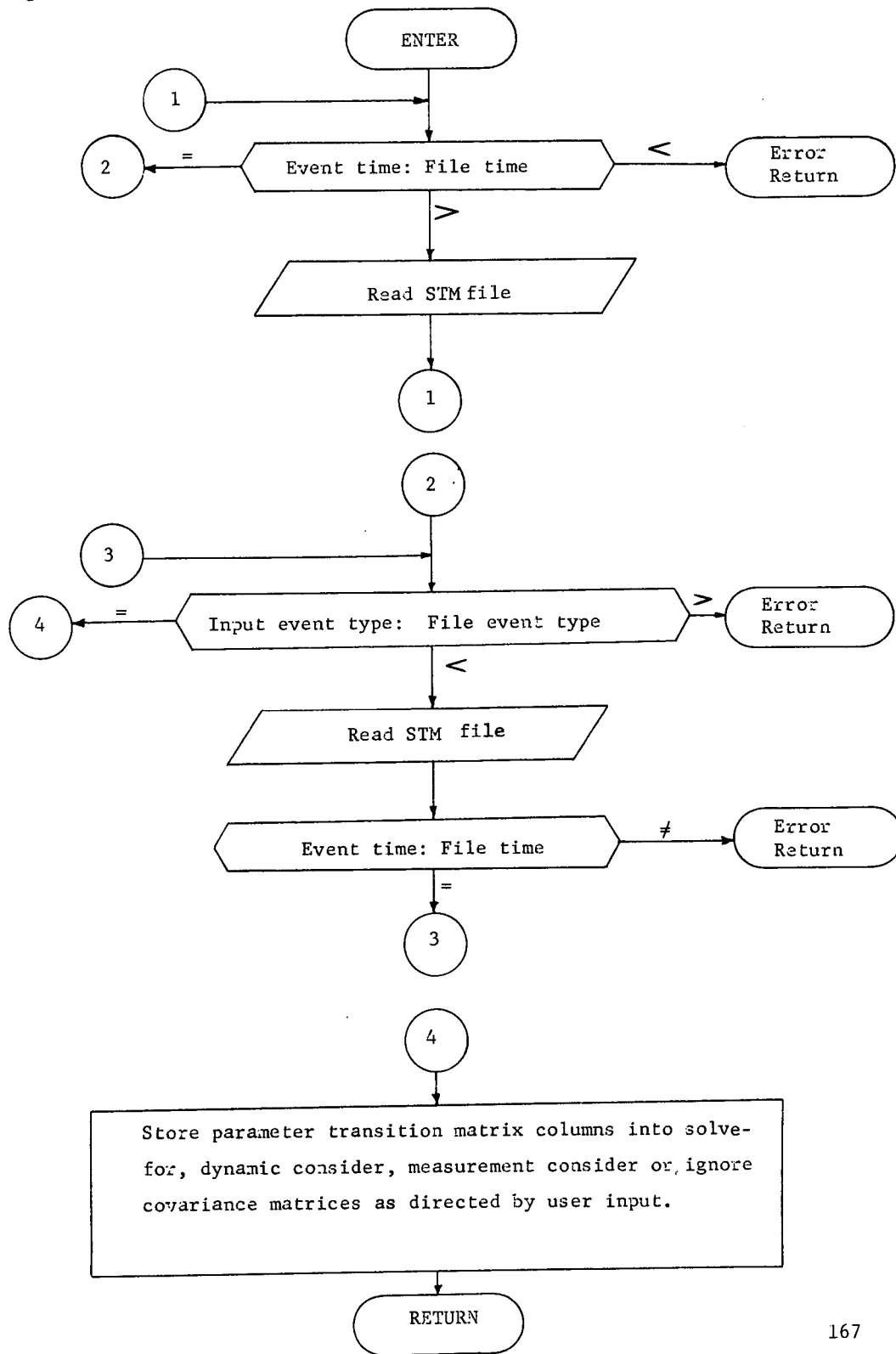
5.3.19 Subroutine STMRDR

Purpose: To read the STM file and prepare information for the error analysis module

Input: o event time
o event type

Output o integrated state vector
o state transition matrix to event time
o parameter transition matrix to event time
o integrated spacecraft variables

Remarks: see subroutine STMGEN for STM file layout



5.3.20 Subroutine TARPRL

Purpose: To compute the partials of the spacecraft state
WRT orbital elements.

Input: o orbital elements, (a,e,i,Ω,ω,M)

Output: o partials of state WRT orbital elements

Remarks:

$$\text{since} \quad \tan \frac{v}{2} = \left[\frac{1+e}{1-e} \right]^{\frac{1}{2}} \tan \frac{E}{2}$$

$$M = E - e \sin E$$

true anomaly can be expressed as a function

$$v = v(e, M)$$

The evaluation of the desired partials can now proceed. The results
are summarized below.

a. Partial with respect to a.

$$\frac{\partial x}{\partial a} = \frac{x}{a}$$

$$\frac{\partial y}{\partial a} = \frac{y}{a}$$

$$\frac{\partial z}{\partial a} = \frac{z}{a}$$

b. Partial with respect to e.

$$\frac{\partial x}{\partial e} = \frac{xq}{r} + r \frac{\partial v}{\partial e} \left[-\cos \Omega \sin(\omega+v) - \sin \Omega \cos(\omega+v) \cos i \right]$$

$$\frac{\partial y}{\partial e} = \frac{yq}{r} + r \frac{\partial v}{\partial e} \left[-\sin \Omega \sin(\omega+v) + \cos \Omega \cos(\omega+v) \cos i \right]$$

$$\frac{\partial z}{\partial e} = \frac{zq}{r} + r \frac{\partial v}{\partial e} \cos(\omega+v) \sin i$$

$$\text{where} \quad q = \frac{r}{ae(1-e^2)} \left[r - a - ae^2(1 + \sin^2 v) \right]$$

c. Partial with respect to i .

$$\frac{\partial x}{\partial i} = r \sin \Omega \sin(\omega + \nu) \sin i$$

$$\frac{\partial y}{\partial i} = - r \cos \Omega \sin(\omega + \nu) \sin i$$

$$\frac{\partial z}{\partial i} = r \sin(\omega + \nu) \cos i$$

d. Partial with respect to Ω .

$$\frac{\partial x}{\partial \Omega} = - y$$

$$\frac{\partial y}{\partial \Omega} = x$$

$$\frac{\partial z}{\partial \Omega} = 0$$

e. Partial with respect to ω .

$$\frac{\partial x}{\partial \omega} = r \left[- \cos \Omega \sin(\omega + \nu) - \sin \Omega \cos(\omega + \nu) \cos i \right]$$

$$\frac{\partial y}{\partial \omega} = r \left[- \sin \Omega \sin(\omega + \nu) + \cos \Omega \cos(\omega + \nu) \cos i \right]$$

$$\frac{\partial z}{\partial \omega} = r \cos(\omega + \nu) \sin i$$

f. Partial with respect to M .

$$\frac{\partial x}{\partial M} = \frac{xs}{r} + r \frac{\partial \nu}{\partial M} \left[- \cos \Omega \sin(\omega + \nu) - \sin \Omega \cos(\omega + \nu) \cos i \right]$$

$$\frac{\partial y}{\partial M} = \frac{ys}{r} + r \frac{\partial \nu}{\partial M} \left[- \sin \Omega \sin(\omega + \nu) + \cos \Omega \cos(\omega + \nu) \cos i \right]$$

$$\frac{\partial z}{\partial M} = \frac{zs}{r} + r \frac{\partial \nu}{\partial M} \cos(\omega + \nu) \sin i$$

where

$$s = \frac{ae \sin \nu}{[1 - e^2]^{\frac{1}{2}}}$$

5.3.21 Program: TEAM

Purpose: To control the execution of the error analysis module.

Input: see Error Analysis Functional I/O Section 3.2

Output: see Error Analysis Functionla I/O Section 3.2

Remarks:

TEAM performs only control logic functions. All analytic functions of the error analysis module are performed by routines subordinate to program TEAM.

Logic flow: see Macrologic, Error Analysis Section 4.1.2

5.3.22 Subroutine TRAKM

Purpose: To compute sensitivities of current measurement type to the state and all parameters.

Input:

- o measurement code
- o spacecraft state vector, x
- o parameter list

Output:

- o observation matrices, $(H_x, H_s, H_u, H_v, H_w)$

Remarks:

Data types available

- o earth based tracking
 - oo 2-way range
 - oo 2-way doppler (range-rate)
 - oo 3-way range
 - oo 3-way doppler
 - oo differenced 2-way and 3-way range
 - oo differenced 2-way and 3-way doppler
 - oo azimuth and elevation angles
- o spacecraft based tracking
 - oo star-planet/target body angles
 - oo planet limb angles (apparent planet diameter)

All Earth-based data types are usable for near Earth missions, and can be taken from any tracking stations desired by the user. Interplanetary missions use all Earth-based data types except azimuth and elevation angles. These types must also be taken only from Deep Space Network (DSN) stations. Nominally stored in the program are the locations for DSN stations Goldstone, Madrid, and Canberra, but these locations may be changed or others added up to a maximum total of nine stations. Spacecraft-based tracking is restricted to interplanetary missions.

Given the measurement model

$$\underline{y} = \underline{h}(\underline{x}, \underline{s}, \underline{u}, \underline{v}, \underline{w})$$

assuming linearity for small deviations from nominal:

$$\delta \underline{y} = H_{\underline{x}} \delta \underline{x} + H_{\underline{s}} \delta \underline{s} + H_{\underline{u}} \delta \underline{u} + H_{\underline{v}} \delta \underline{v} + H_{\underline{w}} \delta \underline{w}$$

where $H_{\underline{x}} = \frac{\partial \underline{h}}{\partial \underline{x}}$, $H_{\underline{s}} = \frac{\partial \underline{h}}{\partial \underline{s}}$, etc.

The Earth-based data types are modeled using the following definitions (see Figure 1 for geometry)

$\underline{r}_h, \dot{\underline{r}}_h$	=	S/C heliocentric position and velocity
$\underline{r}_E, \dot{\underline{r}}_E$	=	Earth heliocentric position and velocity
$\underline{r}_1, \dot{\underline{r}}_1$	=	Station 1 geocentric position and velocity
$\underline{r}_2, \dot{\underline{r}}_2$	=	Station 2 geocentric position and velocity
$\underline{\rho}_1, \dot{\underline{\rho}}_1$	=	S/C position and velocity relative to station 1
$\underline{\rho}_2, \dot{\underline{\rho}}_2$	=	S/C position and velocity relative to station 2
$\underline{u}_1, \underline{u}_2$	=	Unit vectors defining direction of S/C from stations 1 and 2 respectively

- $\rho_1, \dot{\rho}_1$ = S/C range and range-rate from station 1
 $\rho_2, \dot{\rho}_2$ = S/C range and range-rate from station 2
 $\underline{s}_1, \underline{s}_2$ = Spherical geocentric coordinates of stations
 1 and 2, $\underline{s} = (R, \theta, \phi)^T$
 \underline{z} = zero vector, 3x1

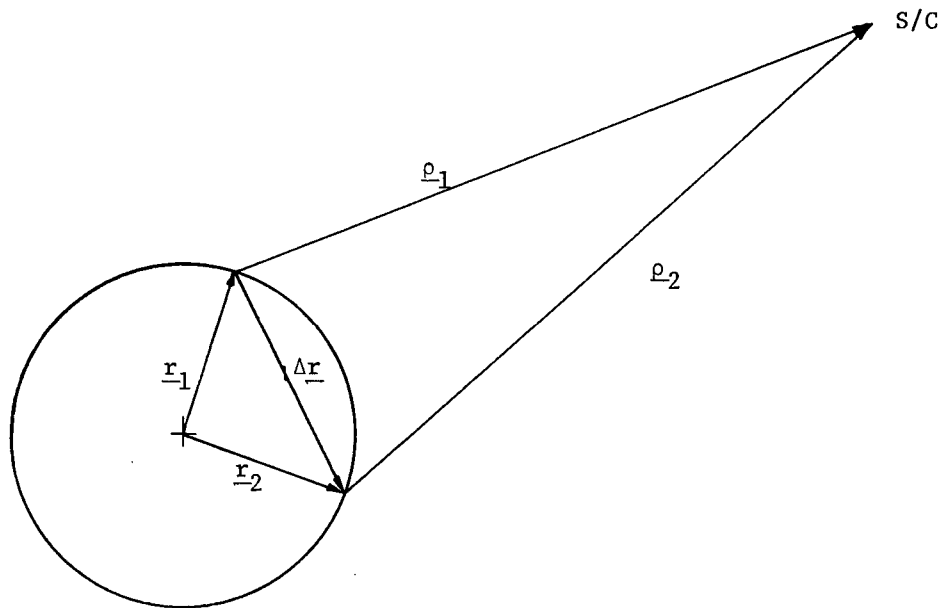


Figure 1: Tracking Geometry for Range and Range-Rate

For two-way tracking the following model and sensitivities result:

$$\rho = |\underline{\rho}| = |\underline{r}_n - \underline{r}_E - \underline{r}_1|$$

$$\dot{\rho} = \dot{\underline{\rho}}^T \cdot \underline{u}$$

$$1) \quad \partial \rho / \partial \underline{x} = \partial \rho / \partial (\underline{r}_n, \dot{\underline{r}}_n) = (\underline{u}_1^T, \underline{z}^T)$$

$$\partial \rho / \partial \underline{s} = \frac{\partial \rho}{\partial (\underline{r}_1, \dot{\underline{r}}_1)} \cdot \frac{\partial (\underline{r}_1, \dot{\underline{r}}_1)}{\partial \underline{s}_1}$$

$$2) \quad \partial \rho / \partial \underline{s} = - \frac{\partial \rho}{\partial \underline{x}} \cdot \frac{\partial (\underline{r}_1, \dot{\underline{r}}_1)}{\partial \underline{s}_1}$$

$$3) \quad \partial \dot{\rho} / \partial \underline{x} = (\partial \dot{\rho} / \partial \underline{r}_n, \partial \dot{\rho} / \partial \dot{\underline{r}}_n)$$

$$4) \quad \partial \dot{\rho} / \partial \underline{r}_n = -\dot{\rho}_1^T \left[\partial \underline{u}_1 / \partial \underline{r}_1 \right]$$

$$5) \quad \partial \dot{\rho} / \partial \dot{\underline{r}}_n = \underline{u}_1^T$$

$$\partial \dot{\rho} / \partial \underline{s} = \frac{\partial \dot{\rho}}{\partial (\underline{r}_1, \dot{\underline{r}}_1)} \cdot \frac{\partial (\underline{r}_1, \dot{\underline{r}}_1)}{\partial \underline{s}_1}$$

$$6) \quad \partial \dot{\rho} / \partial \underline{s} = - \frac{\partial \dot{\rho}}{\partial \underline{x}} \cdot \frac{\partial (\underline{r}_1, \dot{\underline{r}}_1)}{\partial \underline{s}_1}$$

For use in (4) above

$$7) \quad \partial \underline{u}_1 / \partial \underline{r}_1 = \frac{1}{\rho_1} \left[\underline{u}_1 \underline{u}_1^T - \underline{I}_{3 \times 3} \right]$$

Both data types also have bias terms:

$$\rho = |\underline{\rho}| + b_p$$

$$\rho = \underline{\rho}^T \cdot \underline{u} + b_p$$

$$8) \quad \partial \rho / \partial b_p = \partial \dot{\rho} / \partial b_{\dot{p}} = 1$$

Observation types including three-way data, whether as is, or in differencing, are also known as QVLBI (quasi-very long baseline interferometry) data types. Three way data types are modeled as the sum of the two way types plus a timing error term for ranging and a frequency bias term for range-rate.

$$9) \quad \rho_3 = \rho_1 + \rho_2 + c\Delta t$$

$$10) \quad \dot{\rho}_3 = \dot{\rho}_1 + \dot{\rho}_2 + c \frac{\Delta f}{f}$$

where Δt is the timing error, c the speed of light, and $\Delta f/f$ the frequency bias term which results from drift error between the frequency standards at the two separate tracking stations. The sensitivity partials for the three way data types are formed by adding the partials computed for each station individually. The $c \Delta t$ and $c \Delta f/f$ terms are treated either as biases or part of the white noise term. The differenced data types are modeled:

$$11) \quad \Delta \rho = \rho_1 - \rho_2 - c\Delta t$$

$$12) \quad \Delta \dot{\rho} = \dot{\rho}_1 - \dot{\rho}_2 - c \Delta f/f$$

The partials for the differenced data types are formed by differencing the individual partials, with the following exception. Since

$$13) \quad \partial \Delta \rho / \partial \underline{r}_n = \partial \Delta \rho / \partial \underline{r}_n = \left[\underline{u}_1 - \underline{u}_2 \right]^T$$

and \underline{u}_1 and \underline{u}_2 are very nearly equal (as are ρ_1 and ρ_2) for interplanetary missions, we use the following substitutions:

$$14) \quad \Delta \underline{r} = \underline{r}_2 - \underline{r}_1$$

$$15) \quad \Delta \rho = \frac{\left[\underline{u}_1 + \underline{u}_2 \right]^T \cdot \Delta \underline{r}}{1 + \underline{u}_1^T \cdot \underline{u}_2}$$

$$16) \quad \underline{u}_1 - \underline{u}_2 = \left[\Delta \underline{r} - \Delta \rho \underline{u}_2 \right] / \rho_1$$

Spacecraft elevation is computed from

$$17) \quad \beta = \sin^{-1} \left[\underline{u}_1^T \underline{r}_1 / |\underline{r}_1| \right]$$

If elevation is negative, a note is made to that effect on the output file, but the error is not fatal.

For azimuth and elevation angle partials, since no velocity dependence occur, we let

\underline{x} = geocentric ecliptic S/C position

\underline{x}_s = geocentric ecliptic station position

\underline{u}_s = unit vector in \underline{x}_s direction

\underline{w} = unit vector orthogonal to \underline{x}_s and geocentric ecliptic axis

α = S/C azimuth, measured positive from north toward east (see Fig. 2)

β = S/C elevation

$\underline{\rho}$ = S/C range vector from station

\underline{u} = unit vector in $\underline{\rho}$ direction

\underline{x}_a = projection of $\underline{\rho}$ into plane normal to \underline{x}_s

\underline{u}_a = unit vector in direction of \underline{x}_a

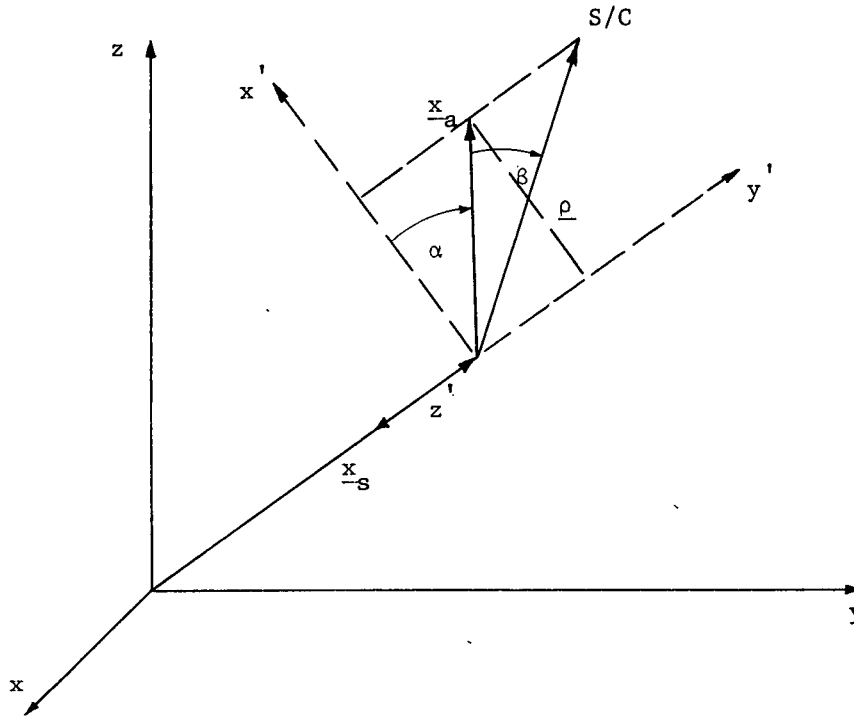


Figure 2: Tracking Geometry for Azimuth and Elevation

The elevation partials are shown first, because they are simpler, and some of them are needed for the azimuth partials.

$$18) \quad \sin \beta = \underline{u}^T \underline{u}_s = \underline{u}_s^T \underline{u}$$

$$19) \quad \cos \beta \frac{\partial \beta}{\partial \underline{x}} = \underline{u}_s^T \frac{\partial \underline{u}}{\partial \underline{x}}$$

$$20) \quad \cos \beta \frac{\partial \beta}{\partial \underline{x}_s} = \underline{u}^T \frac{\partial \underline{u}_s}{\partial \underline{x}_s} + \underline{u}_s^T \frac{\partial \underline{u}}{\partial \underline{x}_s}$$

$$21) \quad \underline{x}_a = \underline{x}_s + \rho \sin \beta \underline{u}_s$$

$$22) \quad \underline{u}_a = \underline{x}_a / |\underline{x}_a|, \quad \underline{w} = \frac{1}{(x_s^2 + y_s^2)^{1/2}} \begin{bmatrix} -y_s, x_s, 0 \end{bmatrix}^T$$

$$23) \quad \sin \alpha = \underline{u}_a^T \underline{w}$$

$$24) \quad \cos \alpha \partial \alpha / \partial \underline{x} = \underline{w}^T \partial \underline{u}_a / \partial \underline{x}$$

$$25) \quad \partial \underline{u}_a / \partial \underline{x} = \frac{\partial \underline{u}_a}{\partial \underline{x}_a} \cdot \frac{\partial \underline{x}_a}{\partial \underline{x}}$$

$$26) \quad \frac{\partial \underline{x}_a}{\partial \underline{x}} = \sin \beta \underline{u}_s \frac{\partial \rho}{\partial \underline{x}} + \rho \cos \beta \underline{u}_s \partial \beta / \partial \underline{x}$$

$$27) \quad \cos \alpha \partial \alpha / \partial \underline{x}_s = \underline{w}^T \partial \underline{u}_a / \partial \underline{x}_s + \underline{u}_a^T \partial \underline{w} / \partial \underline{x}_s$$

$$28) \quad \partial \underline{u}_a / \partial \underline{x}_s = \frac{\partial \underline{u}_a}{\partial \underline{x}_a} \cdot \frac{\partial \underline{x}_a}{\partial \underline{x}_s}$$

$$29) \quad \partial \underline{x}_a / \partial \underline{x}_s = I_{3 \times 3} + \rho \sin \beta \partial \underline{u}_s / \partial \underline{x}_s + \sin \beta \underline{u}_s \partial \rho / \partial \underline{x}_s + \rho \cos \beta \underline{u}_s \partial \beta / \partial \underline{x}_s$$

$$30) \quad \partial \underline{w} / \partial \underline{x}_s = \begin{bmatrix} w_1 & w_2 & w_2^2 - 1 & 0 \\ 1 - w_1^2 & -w_1 w_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \frac{1}{(x_s^2 + y_s^2)^{1/2}}$$

To complete the station location partials we have

$$31) \quad \partial(\alpha, \beta) / \partial \underline{s} = \frac{\partial(\alpha, \beta)}{\underline{x}_s} \cdot \frac{\partial \underline{x}_s}{\partial \underline{s}}$$

This is the same form used for the range and range-rate partials, where

$\partial \underline{x}_s / \partial \underline{s}$ comes from subroutine STAPRL (5.3.16).

For spacecraft based measurements, we use the following definitions:

\underline{x} = spacecraft position vector

\underline{x}_p = planet position vector

$\underline{\rho}$ = $\underline{x}_p - \underline{x}$ = planet range vector

\underline{d} = vector of star direction cosines

\underline{p} = vector of target planet orbital elements

R_p = target planet radius

γ = star-planet angle

δ = apparent planet diameter measurement - angle subtended by planet disc at the spacecraft

\underline{z} = zero vector, 3x1

Star-planet angle:

$$32) \cos \gamma = \underline{d}^T \cdot \underline{u}_p$$

$$33) \frac{\partial \gamma}{\partial \underline{x}} = \frac{1}{\rho \sin \gamma} \left[\underline{d}^T - \underline{u}^T \cos \gamma \right] \frac{\partial \underline{u}}{\partial \underline{x}}$$

$$34) \frac{\partial \gamma}{\partial \dot{\underline{x}}} = \underline{z}^T$$

$$35) \frac{\partial \gamma}{\partial \underline{p}} = \frac{\partial \gamma}{\partial \underline{x}_p} \cdot \frac{\partial \underline{x}_p}{\partial \underline{p}} = - \frac{\partial \gamma}{\partial \underline{x}} \cdot \frac{\partial \underline{x}_p}{\partial \underline{p}}$$

Apparent planet diameter:

$$36) \sin \delta / 2 = R_p / \rho$$

$$37) \frac{\partial \delta}{\partial \underline{x}} = \frac{2 R_p \underline{u}^T}{\rho^2 R_p^{3/2}}$$

$$38) \frac{\partial \delta}{\partial \dot{\underline{x}}} = 0$$

* $\partial \underline{x}_p / \partial \underline{p}$ generated by TARPRL (5.3.20)

$$39) \quad \partial \delta / \partial \underline{p} = \frac{\partial \delta}{\partial \underline{x}_p} \cdot \frac{\partial \underline{x}_p}{\partial \underline{p}} = - \frac{\partial \delta}{\partial \underline{x}} \cdot \frac{\partial \underline{x}_p}{\partial \underline{p}}$$

The sensitivities of all angle measurements to their respective biases are unity, as were the corresponding sensitivities for range and range-rate.

Several places in the foregoing derivations, the partial derivative of a unit vector is needed with respect to its "parent" vector. Therefore the following notations are made. Define \underline{u} as a unit vector in the direction of \underline{a}

$$\underline{u} = \underline{a} / |\underline{a}|$$

then

$$\partial \underline{u} / \partial \underline{a} = \frac{1}{a} \left[\mathbf{I} - \underline{u} \underline{u}^T \right]$$

$$a = |\underline{a}|$$

If

$$\underline{a} = \underline{b} - \underline{c}$$

then

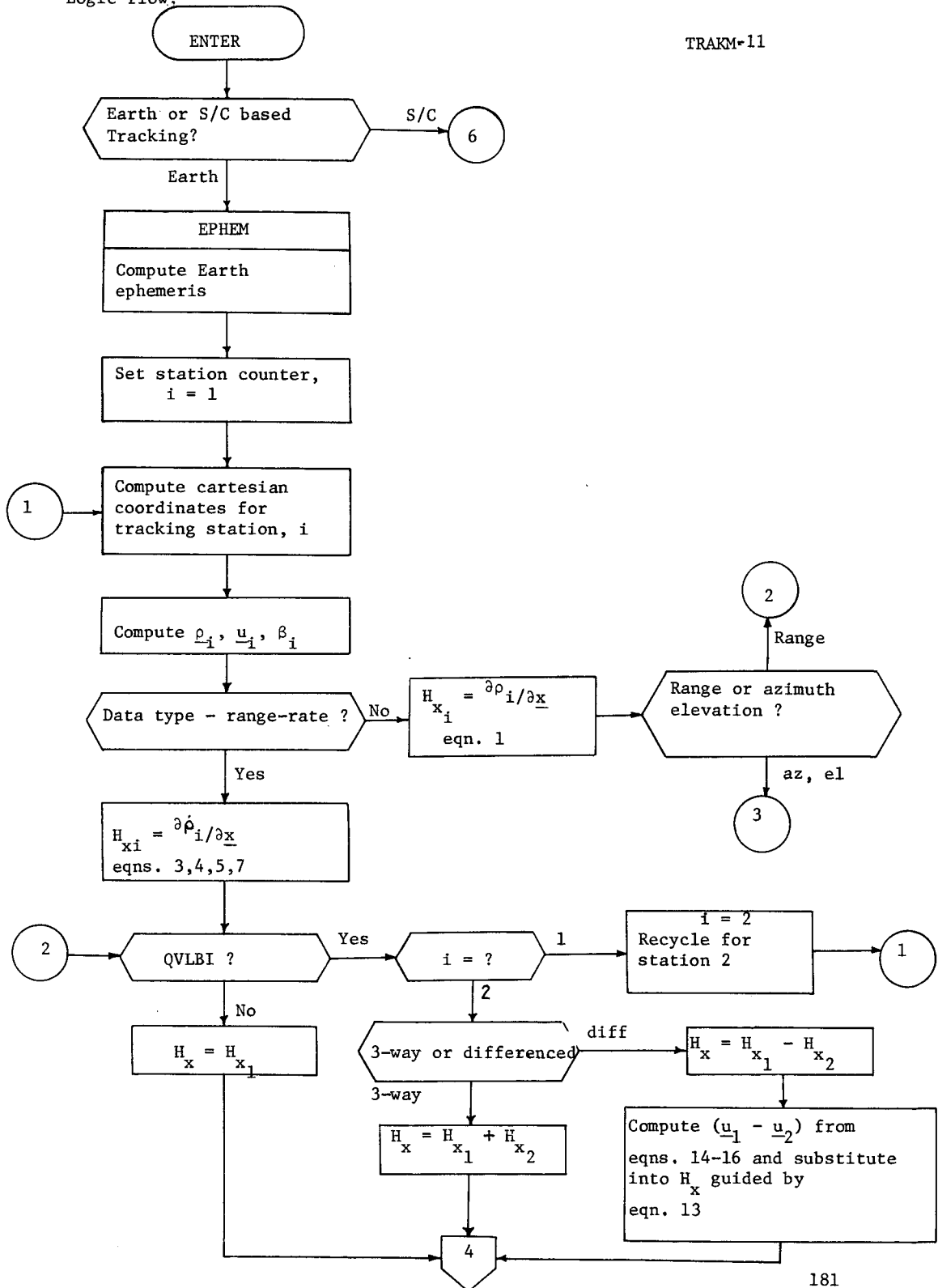
$$\partial \underline{u} / \partial \underline{b} = \partial \underline{u} / \partial \underline{a}$$

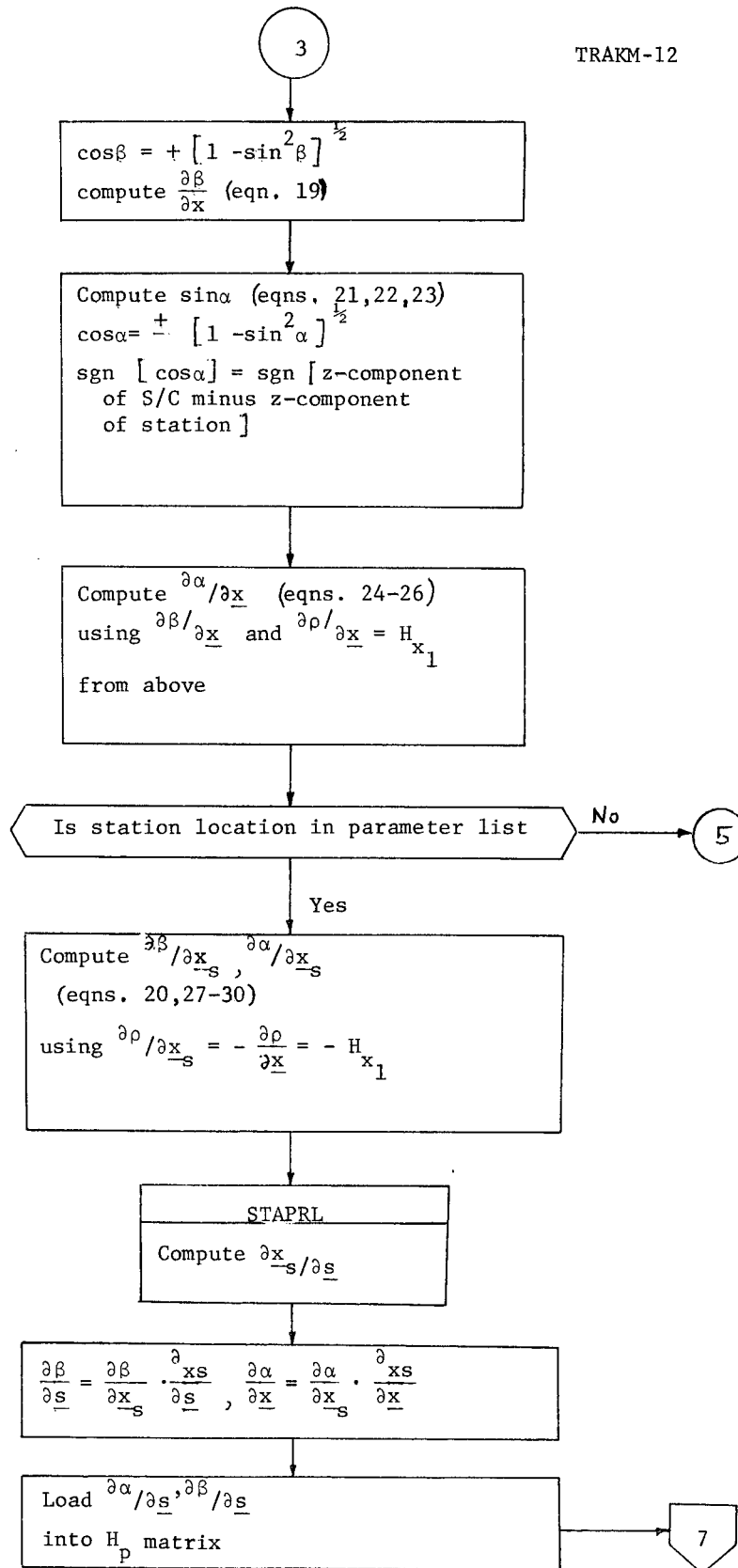
$$\partial \underline{u} / \partial \underline{c} = - \partial \underline{u} / \partial \underline{a}.$$

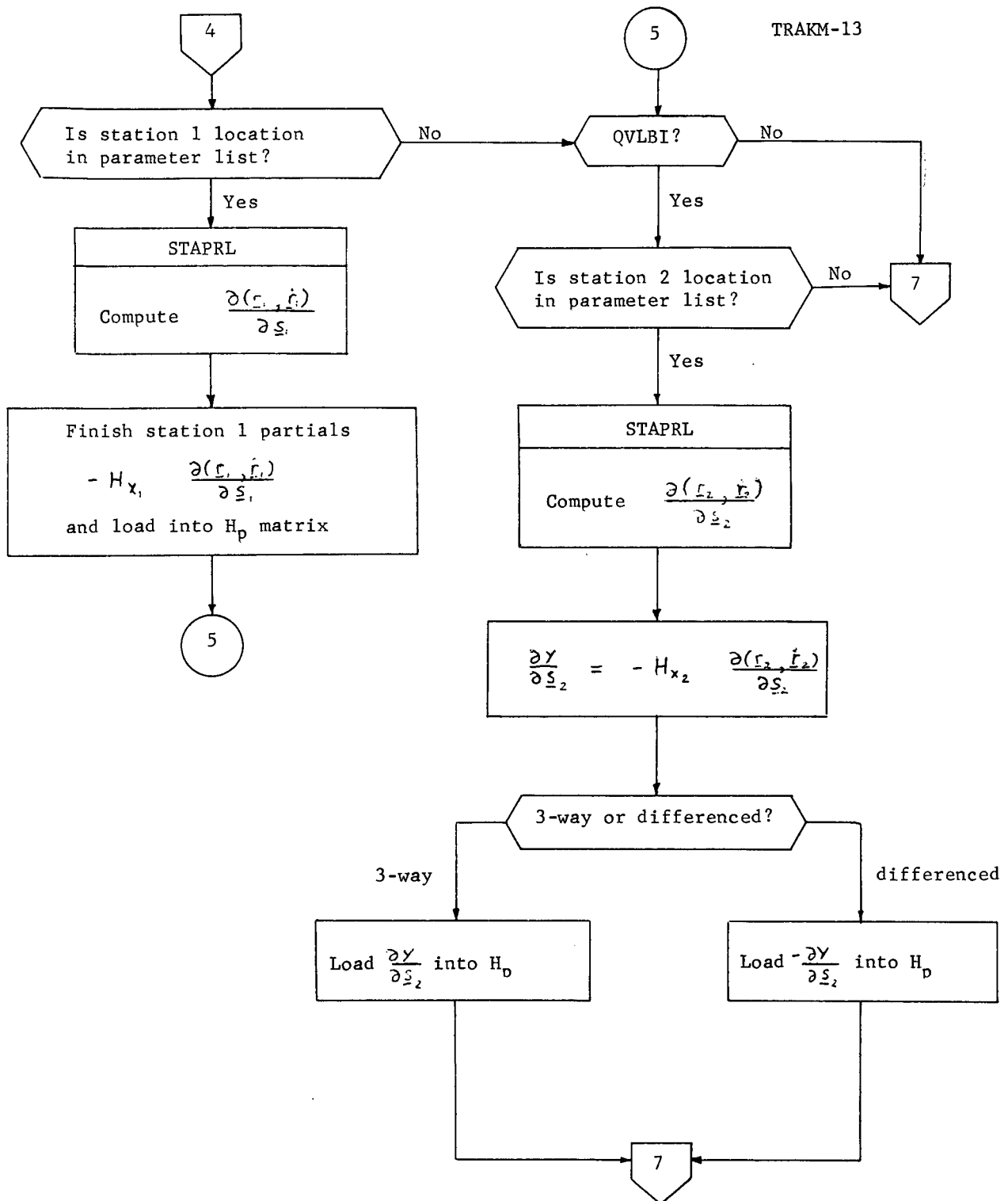
Also, TRAKM is designed to receive a parameter list of all solve-for, dynamic and measurement consider, and ignore parameters. For determining observation sensitivities to parameters, one matrix, H_p is defined initially to include all sensitivities which are then later rearranged column by column into the H_s , H_u , H_v , H_w matrices. This simplifies the logic necessary to compute the observation sensitivities initially.

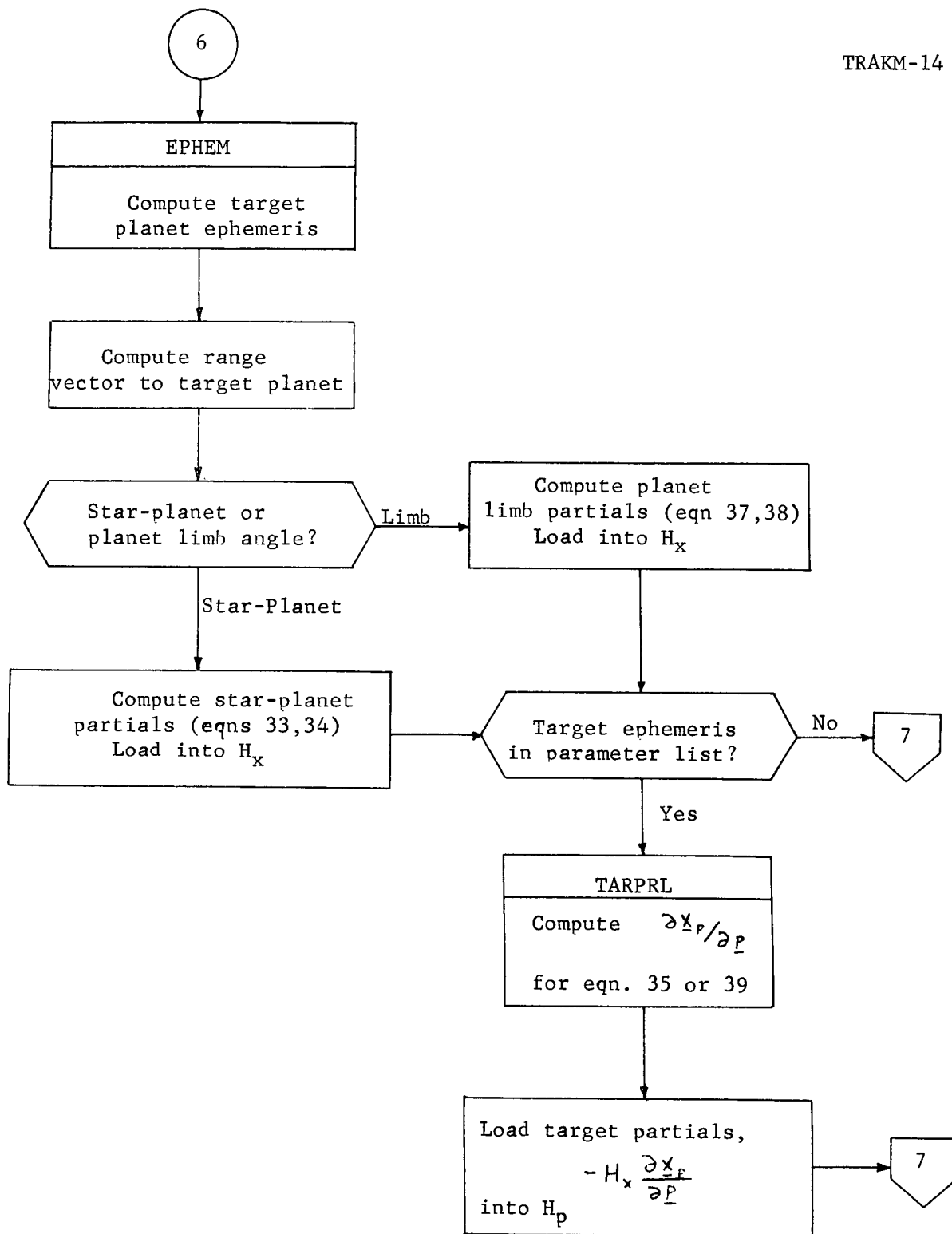
Logic flow;

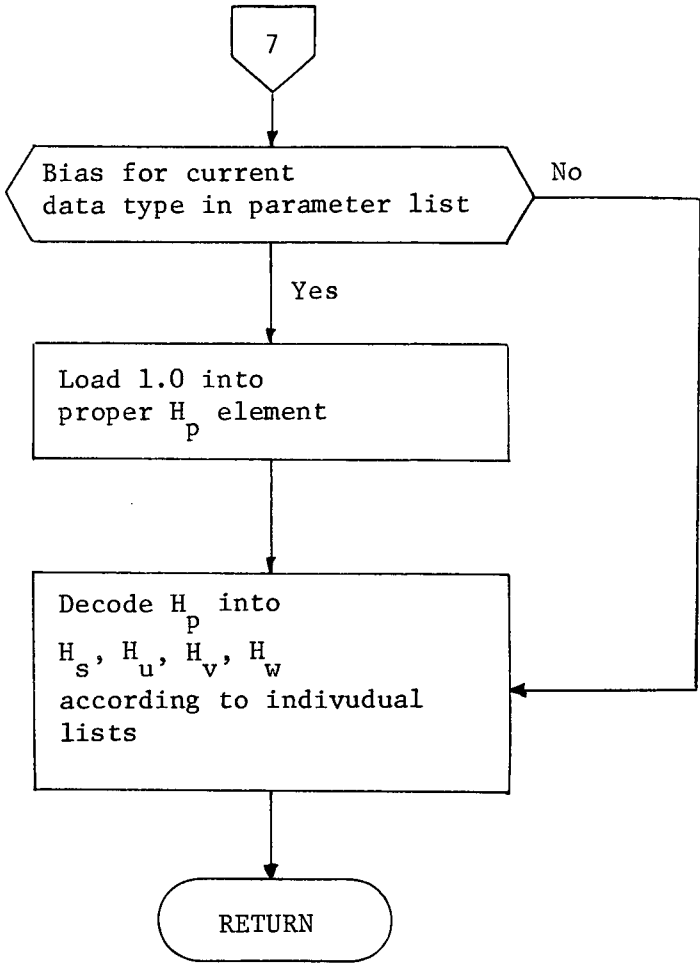
TRAKM-11











5.3.23 Subroutine USRGAN

Purpose: To compute filter gain by user supplied algorithm.

Remarks:

User must supply his own FORTRAN subroutine and see that it is compatible with the calling sequence in subroutine FILTER (5.3.4)

5.3.24 Subroutine WLSGAN

Purpose: To compute gain matrix partitions for sequential weighted least squares filter, and maintain the non-consider covariances necessary for that computation.

Input:

- o flag indicating gain matrix computation, or non-consider covariance propagation
- o for gain matrix computation;
 - oo observation matrices, H_x , H_s
 - oo measurement noise covariance, R
- o for non-consider covariance propagation; state and solve-for transition matrices

Output:

- o gain matrix partitions for state (K_x) and solve-for parameters (K_s)

Remarks:

The sequential, or recursive weighted least squares (WLS) algorithm implemented here is equivalent to a batch WLS filter if there is no process noise. Since process noise is a significant part of low thrust analysis, the WLS filter must be used recursively, because it has no batch equivalent. The sequential WLS consider filter acknowledges consider parameters only for covariance analysis, and not for gain matrix calculation. Therefore a set of "non-consider" covariances for the state and solve-for parameters must be maintained at all times. This set also represents the filter analysis as it would be in non-consider form.

Each time the knowledge covariances are propagated - except for prediction - subroutine PROP (5.3.13) also calls WLSGAN to propagate the

WLS non-consider covariances. At a measurement event WLSGAN computes the gain matrix partitions K_x and K_s , and also updates the non-consider covariances. All covariances in the equations below are non-consider, not knowledge.

Non-consider covariance propagation:

$$1) \quad P^- = \left[\Phi P^+ + \theta_{xs} C_{xs}^{+T} \right] \Phi^T + C_{xs}^- \theta_{xs}^T$$

$$2) \quad C_{xs}^- = \Phi C_{xs}^+ + \theta_{xs} P_s^+$$

$$3) \quad P_s^- = P_s^+$$

Gain matrix computations:

$$4) \quad A = P_x^- H_x^T + C_{xs}^- H_s^T$$

$$5) \quad B = P_s^- H_s^T + C_{xs}^{-T} H_x^T$$

$$6) \quad J = H_x A + H_s B + R$$

$$7) \quad K_x = A J^{-1}$$

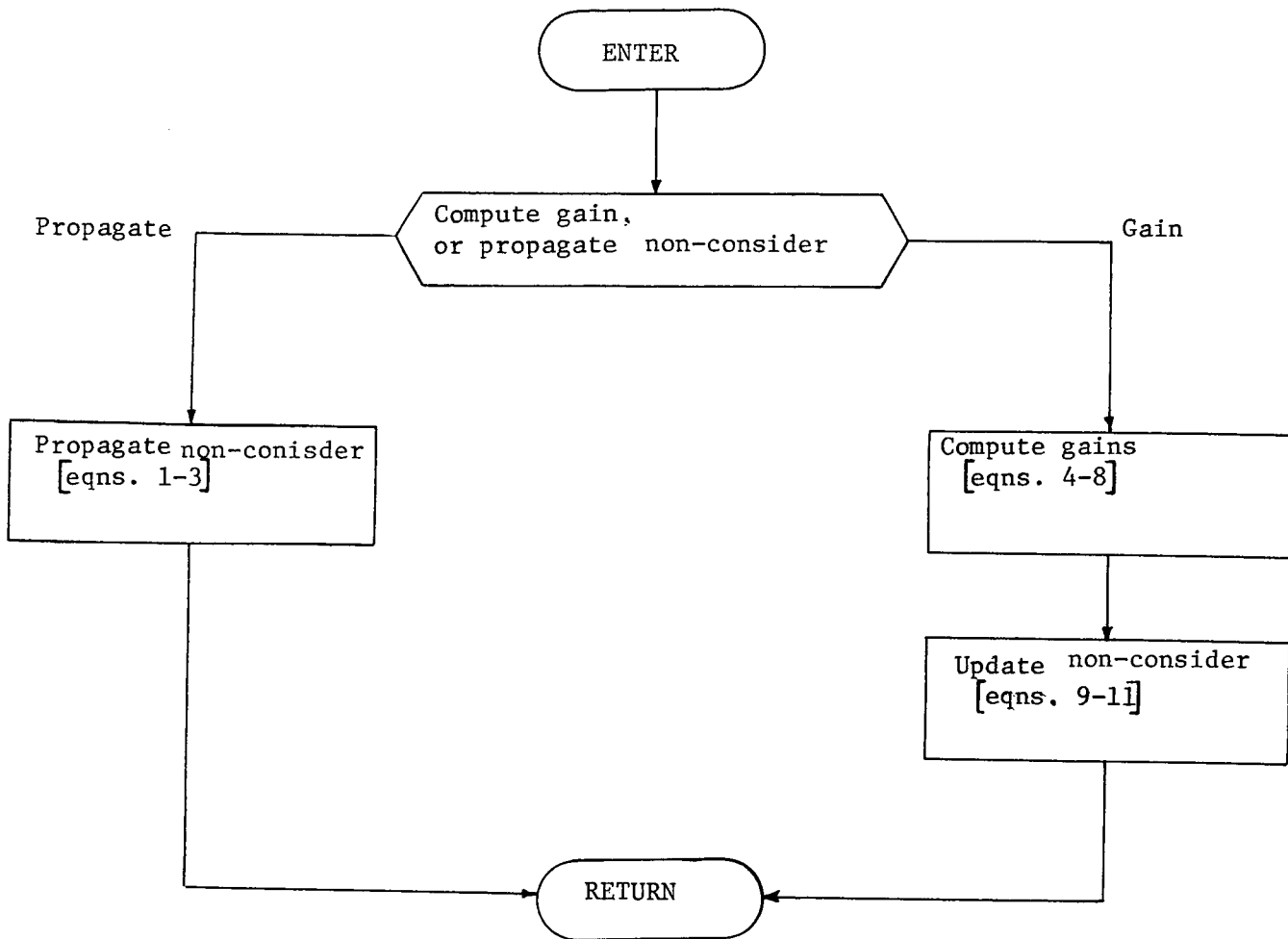
$$8) \quad K_s = B J^{-1}$$

Non-consider covariance update

$$9) \quad P^+ = P^- - K_x A^T$$

$$10) \quad C_{xs}^+ = C_{xs}^- - K_x B^T$$

$$11) \quad P_s^+ = P_s^- - K_s B^T$$



5.3.25 Subroutine XGUID

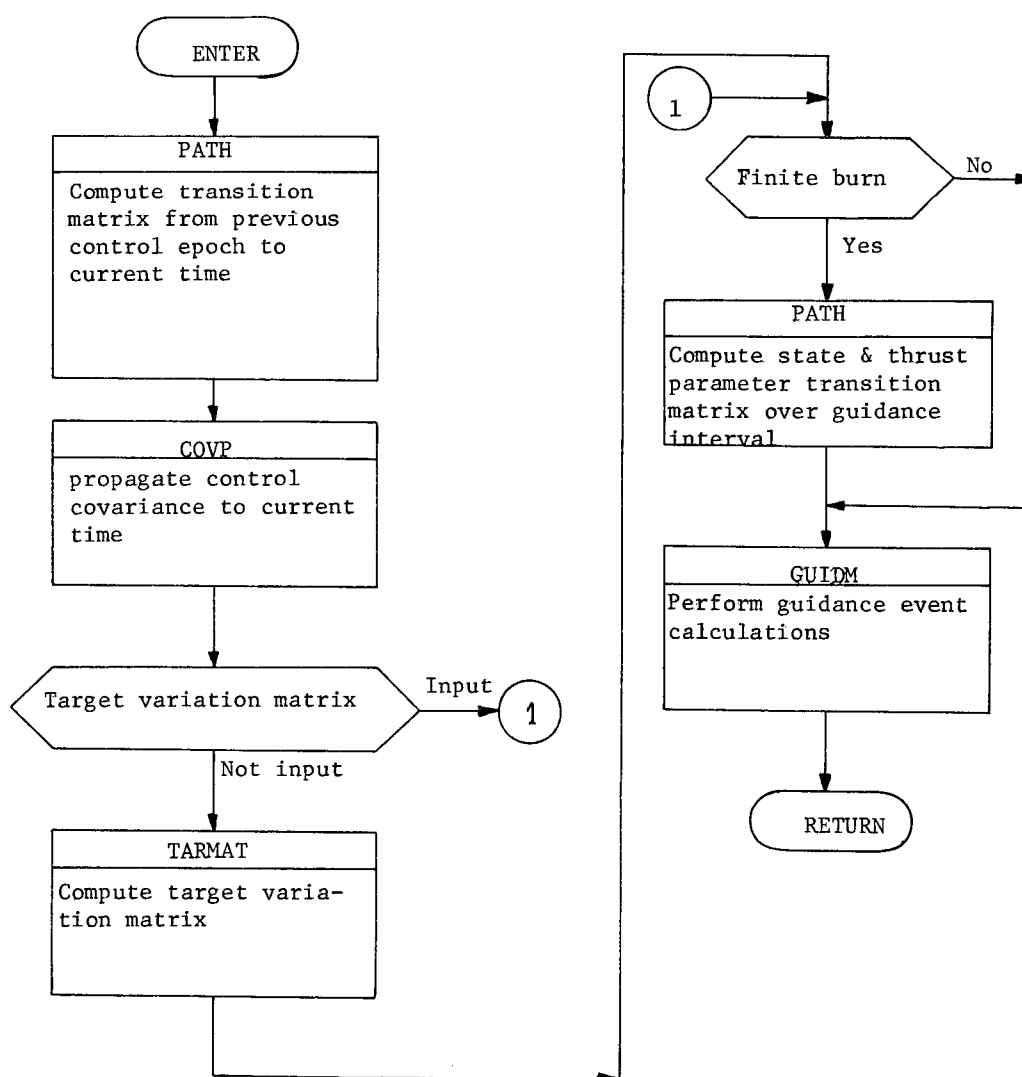
Purpose: To control the execution of a guidance event.

Input:

- current time (guidance epoch)
- time of last control epoch
- guidance cutoff time
- target variation matrix flag
- finite burn flag

Output:

- updated control covariance
- target variation matrix



5.4 Simulation Mode

5.4.1 Subroutine CSAMP

Purpose: To sample eigenvalue of a covariance and rotate back into state space and form a sampled vector

Input:

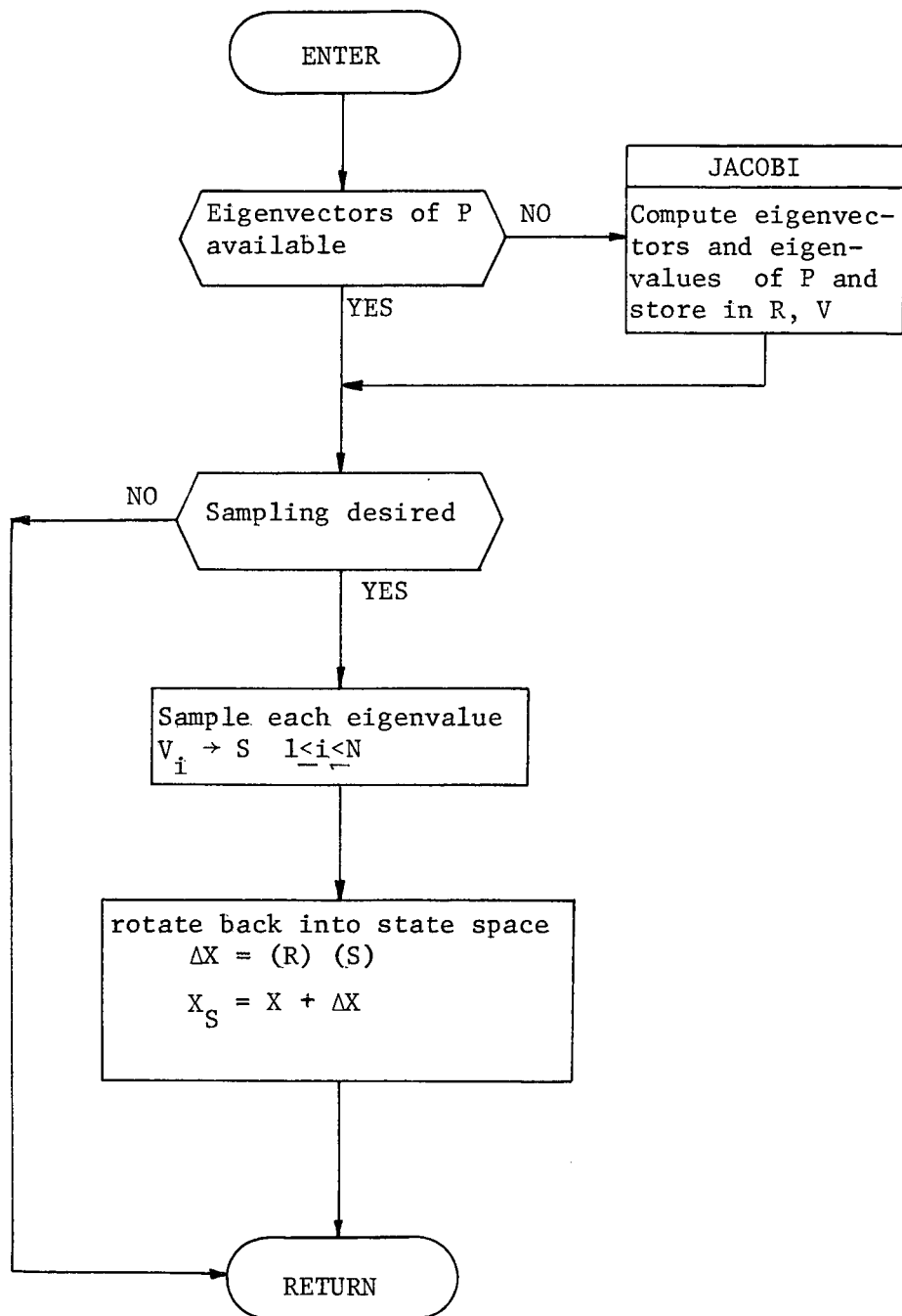
- o state covariance, P
- o reference state, X
- o flag for determining sample option
- o dimension of state covariance, N
- array of eigenvectors (R) and eigenvalues (V)

Output:

- o sampled state vector, X_s
- o array of eigenvectors and eigenvalues, R and V

Remarks:

- o Each eigenvalue is sampled assuming a normal distribution with zero mean using the function RNUM



5.4.2 Subroutine: DATAS

Purpose: To read input and initialize trajectory simulation mode

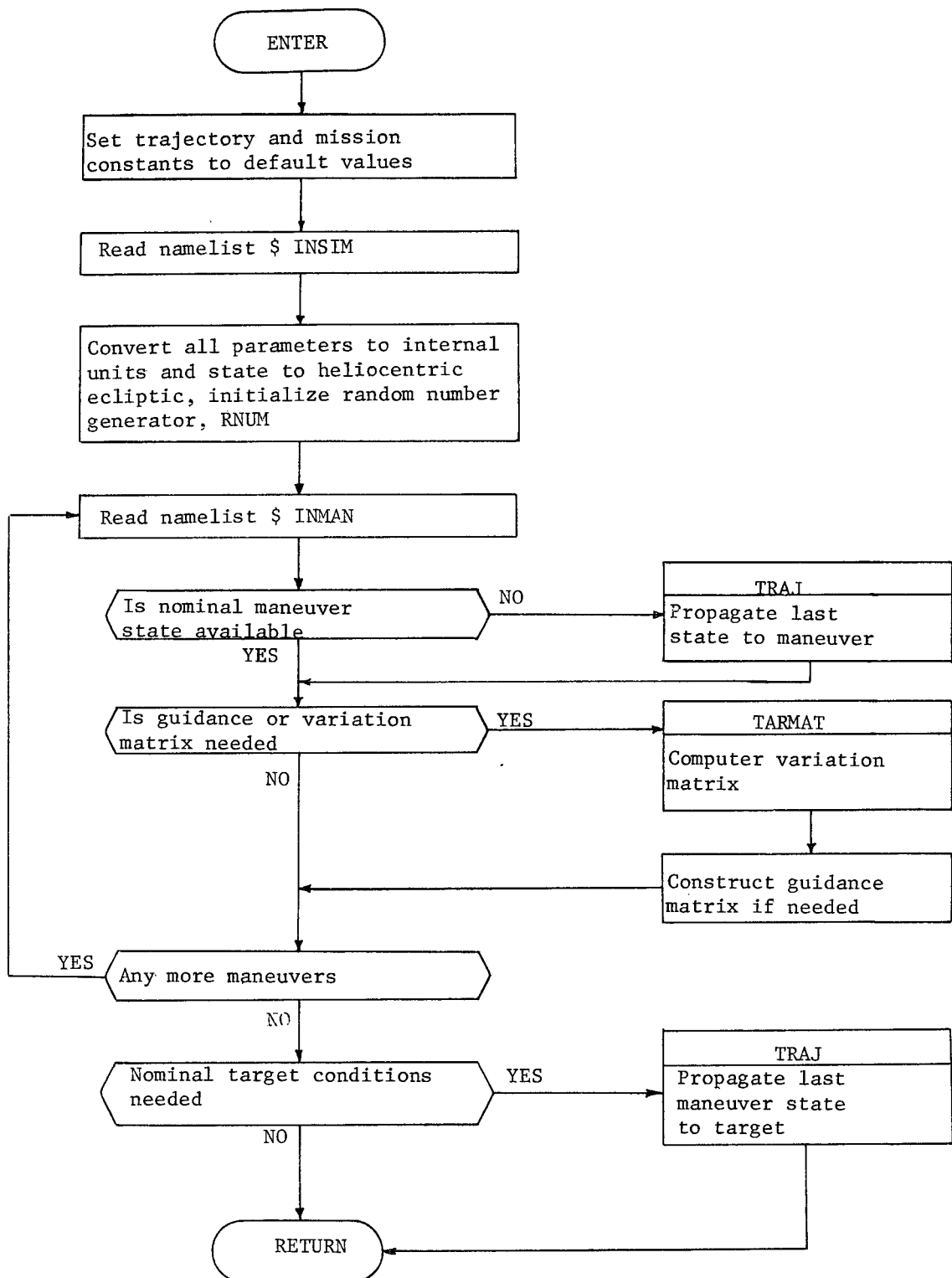
Input: See TSIM Program Description (Section 2.3)

Output:

- o nominal spacecraft state at all maneuver times
- o guidance or variation matrices for all maneuvers
- o nominal target conditons
- o error distributions of trajectory and mission parameters
- o all parameters in proper units and reference frames
- o random number initialization sequence

Remarks:

DATAS will prepare all data for subsequent Monte Carlo operation. User options will specify the degree of data preparation necessary, e.g., whether target variation matrices are input or should be computed and whether a priori error statistics are available from a previous run. Guidance and variation matrices are computed as in Sections 5.3.6 and 5.4.9 respectively.



5.4.3 Subroutine GUIDS

Purpose: To design the control correction necessary for a guidance event in the trajectory simulation mode.

Input:

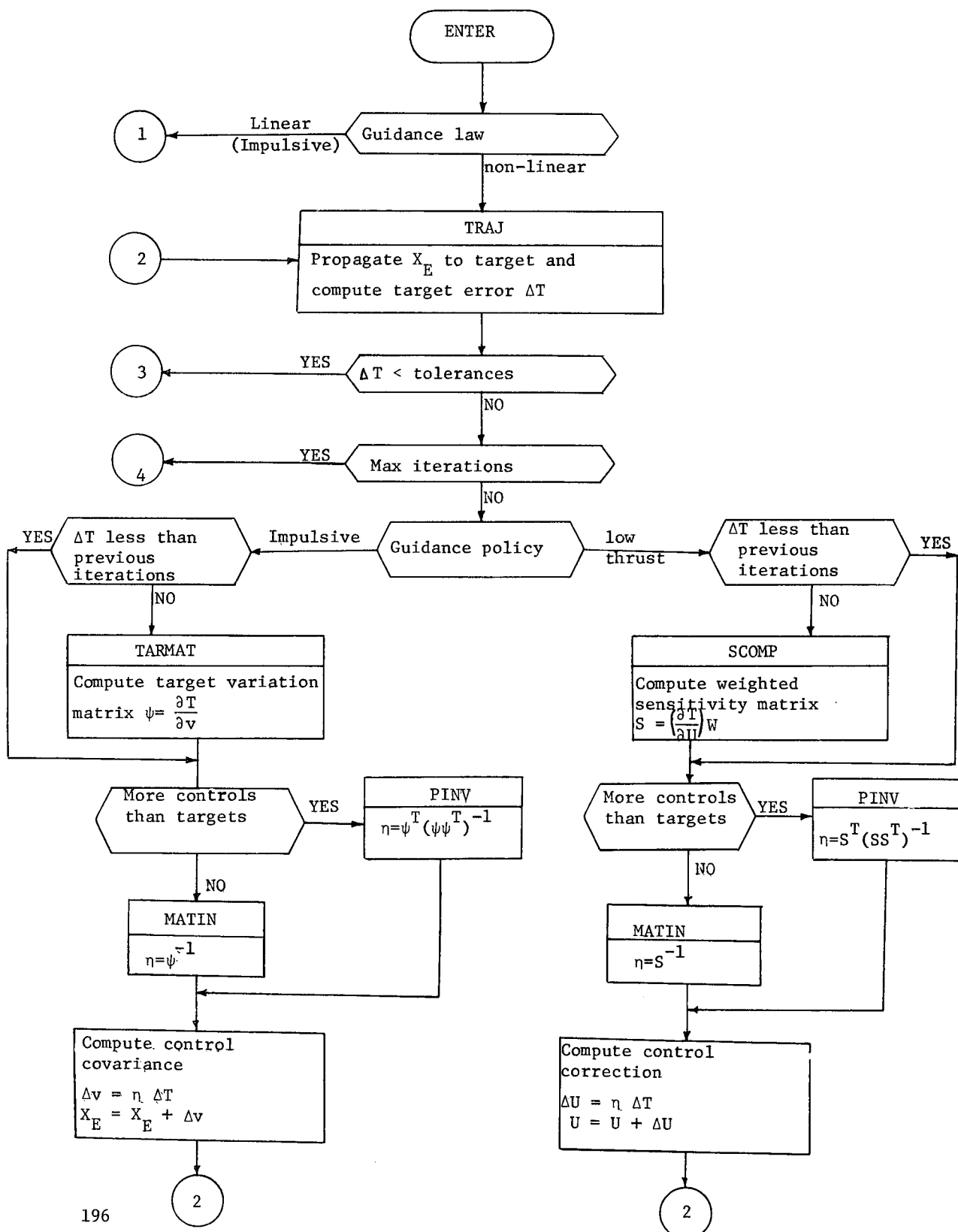
- o maneuver epoch
- o estimated spacecraft state at epoch, X_E
- o reference state at epoch, X_R
- o target body and stopping condition
- o target values and tolerances
- o guidance law: linear or non-linear
- o guidance policy: impulsive or low thrust
- o allowable thrust controls (U) and weighting (W)
- o target variation matrix (ψ) or sensitivity matrix (S)
- o maximum number of iterations
- o guidance matrix (for linear impulsive ΔV)

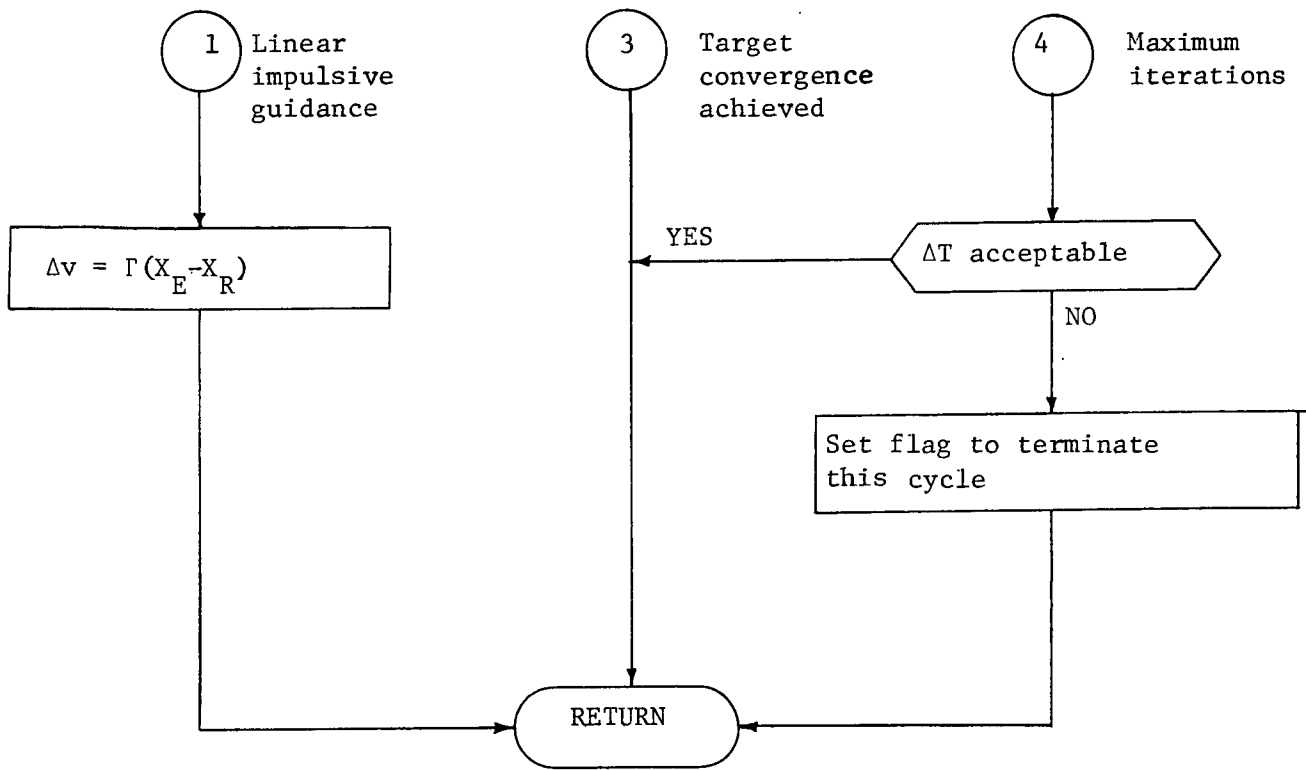
Output:

- o design control correction (Δv or ΔU)
- o estimated target error before and after maneuver
- o target variation or sensitivity matrix
- o cycle termination flag

Remarks:

Non-linear guidance applies a linear algorithm in iterative fashion. A successful maneuver design occurs when the corrected trajectory meets all target conditions within their tolerances. A near successful design occurs when the corrected trajectory comes "close" to meeting the target tolerances. "Close" may be defined as some scaler of the target tolerances. Should the non-linear design sequence exceed the maximum number of iterations and not come close to target tolerances, then the maneuver is deemed hopeless to make, and the mission cycle is terminated. The target matrix (ψ or S) is recomputed only if target error has increased since the last iteration.





5.4.4 Subroutine NOISE

Purpose: To compute thrust acceleration perturbations due to time-varying noise.

Input:

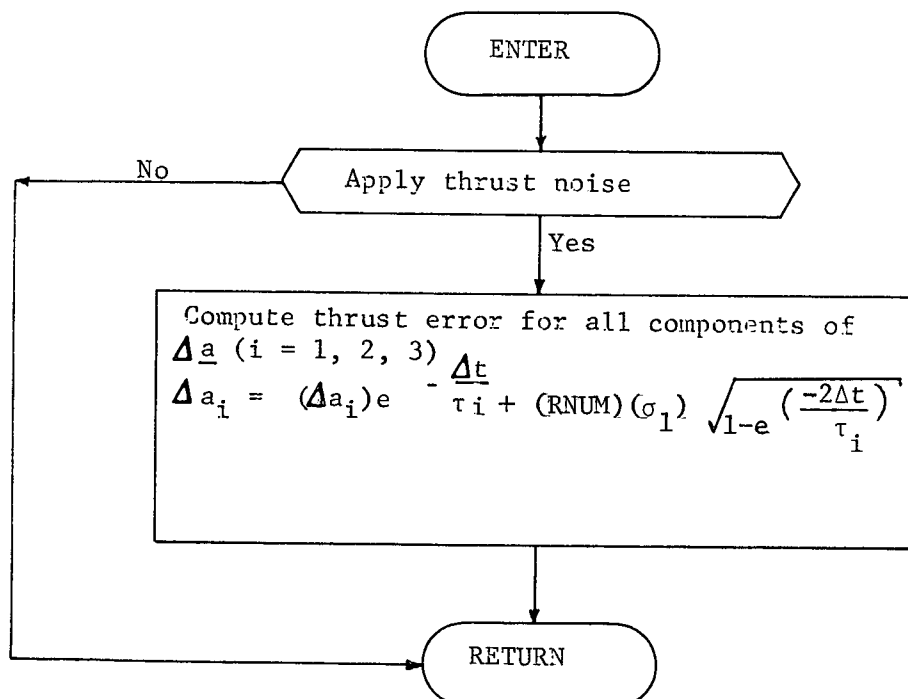
- o standard deviation in thrust proportionality and two pointing angles, σ
- o noise correlation times, τ
- o noise flag (yes or no)
- o time interval, Δt
- o present discrete thrust error, Δa

Output: o new thrust error, Δa

Remarks:

The form of the thrust noise assumes acceleration error components to be independent of each other and to have Gauss-Markov properties. Use is made of the function RNUM to find Gaussian zero-mean, unit variance random numbers.

Logic Flow:



5.4.5 Function RNUM

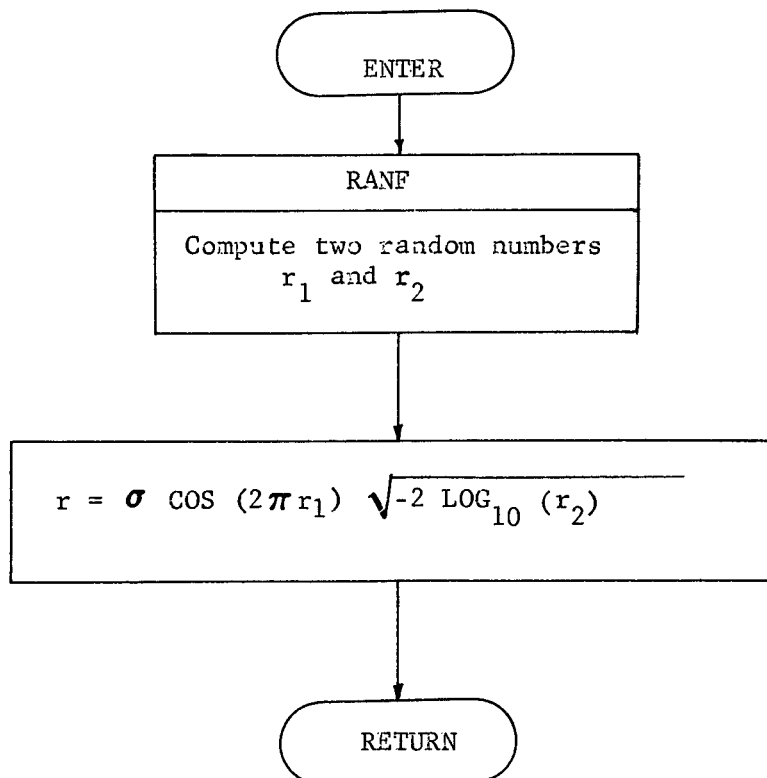
Purpose: To generate a Gaussian zero mean random number.

Input: standard deviation, σ

Output: random number, r

Remarks:

There exist many random number algorithms all of which perform equally well. The method used here requires a CDC system function (RANF) which generates a uniformly distributed random number between ± 1 .



5.4.6 Subroutine SCOMP

Purpose: To compute the sensitivity matrix of target parameters
WRT control parameters.

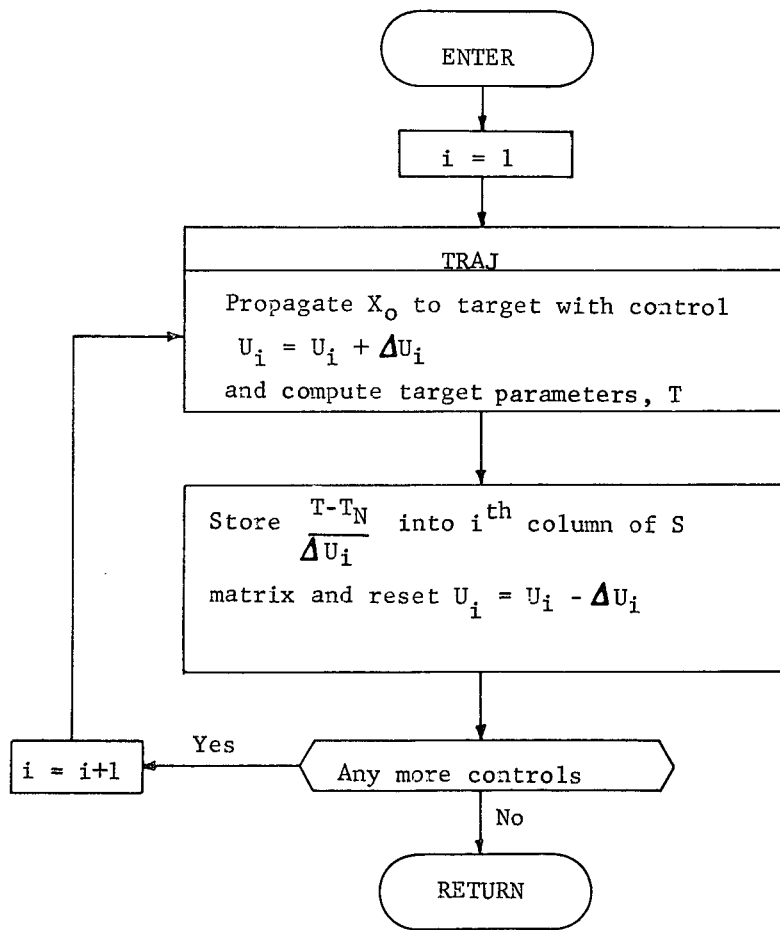
Input:

- o epoch, t_0
- o nominal state, X_0
- o nominal target values, T_N
- o target cutoff condition
- o nominal control parameters, U
- o control perturbations, ΔU

Output: o sensitivity matrix, S

Remarks:

The sensitivity matrix is computed by numerical differencing techniques.



5.4.7 Subroutine SETUP

Purpose: To transfer real-world or nominal or estimated values
 into working arrays.

Input: o ephemeris and gravitational constants
 o spacecraft constants
 o thrust control constants

Output: o ephemeris and gravitational constants
 o spacecraft constants
 o thrust control constants

Remarks:

SETUP is used to store appropriate constants into arrays which are accessed by other routines. For example, prior to designing a maneuver in the simulation mode, SETUP is called to insert nominal mission values so that GUIDS will design the maneuver under the proper assumptions.

Logic Flow: None

5.4.8 Subroutine STAT

Purpose: To compute cumulative mean and covariance of error vector

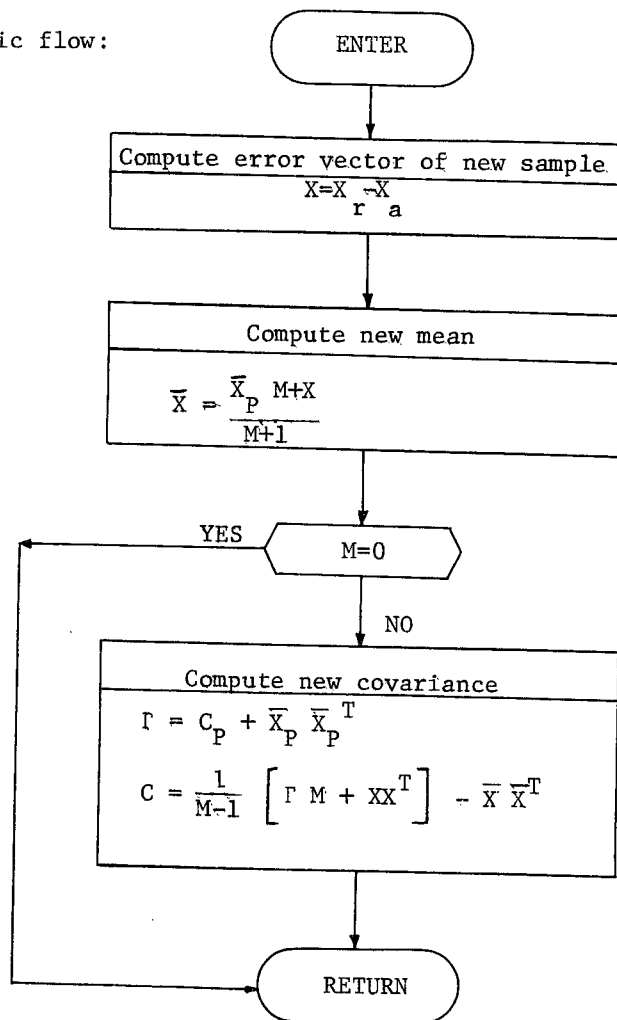
Input:

- o Vector of actual values, X_a
- o vector of reference values, X_r
- o vector dimension, N
- o number of previous samples, M
- o mean (\bar{X}_p) and covariance (C_p) of previous samples

Output:

- o mean (\bar{X}) and covariance (C) of total samples

Logic flow:



5.4.9 Subroutine TARMAT

Purpose: To compute target variation matrix and target parameters.

Input;

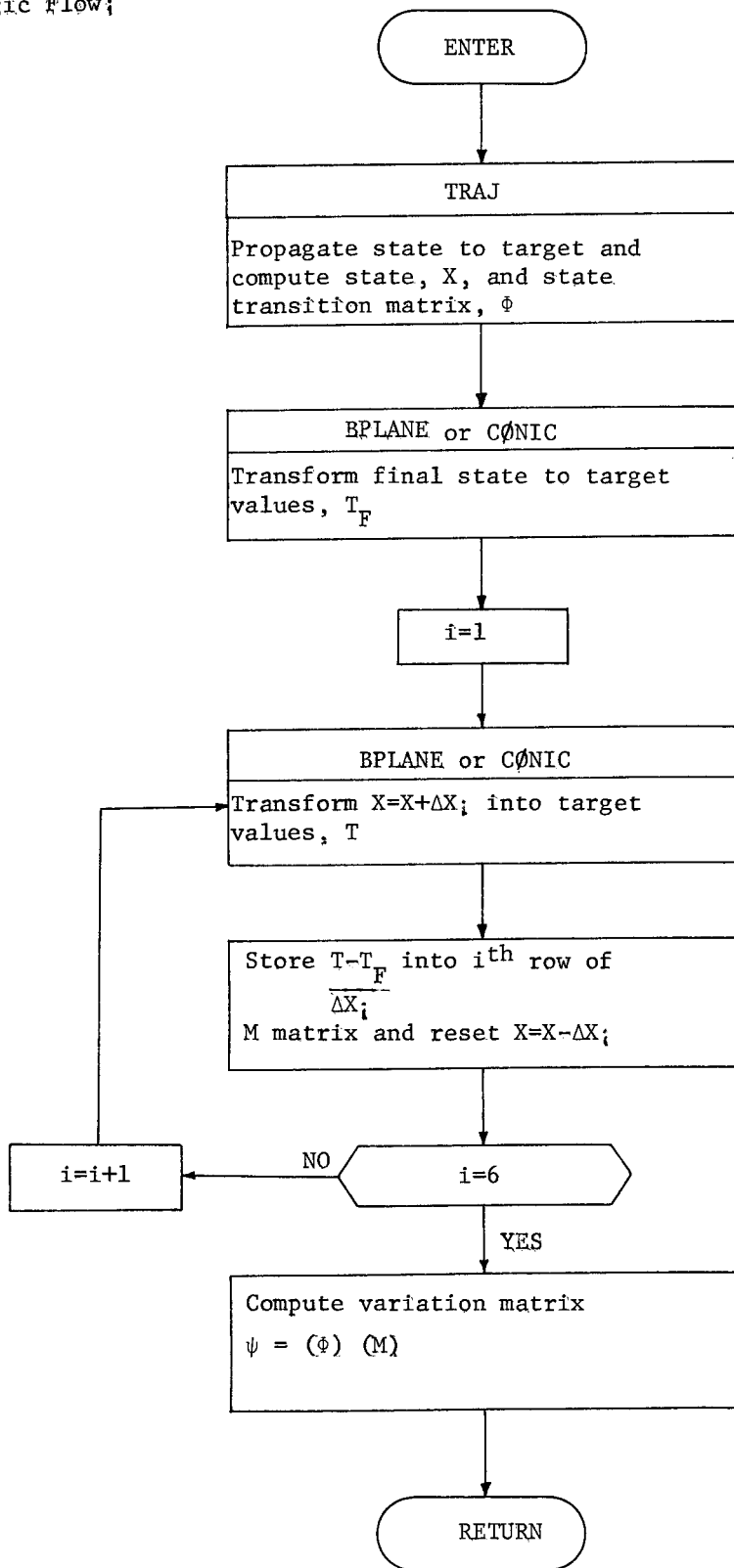
- o initial time
- o initial state and thrust controls
- o target cutoff condition
- o target parameters list
- o state perturbations, ΔX

Output;

- o target parameters, T_F
- o target variation matrix, ψ

Remarks;

The variation matrix is computed using the product of the state transition matrix (from initial to target time) and a target transformation matrix (at target time).



5.4.10 Subroutine TSIM

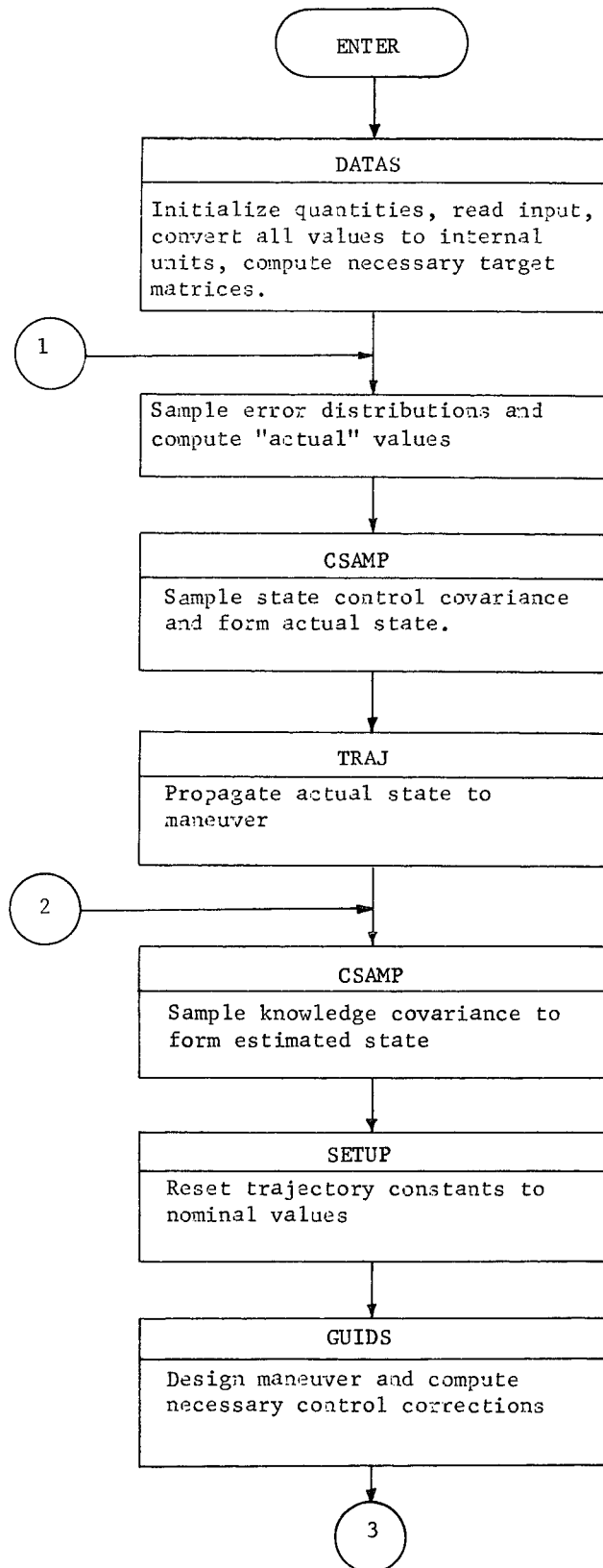
Purpose: To control overall logic of the trajectory simulation mode.

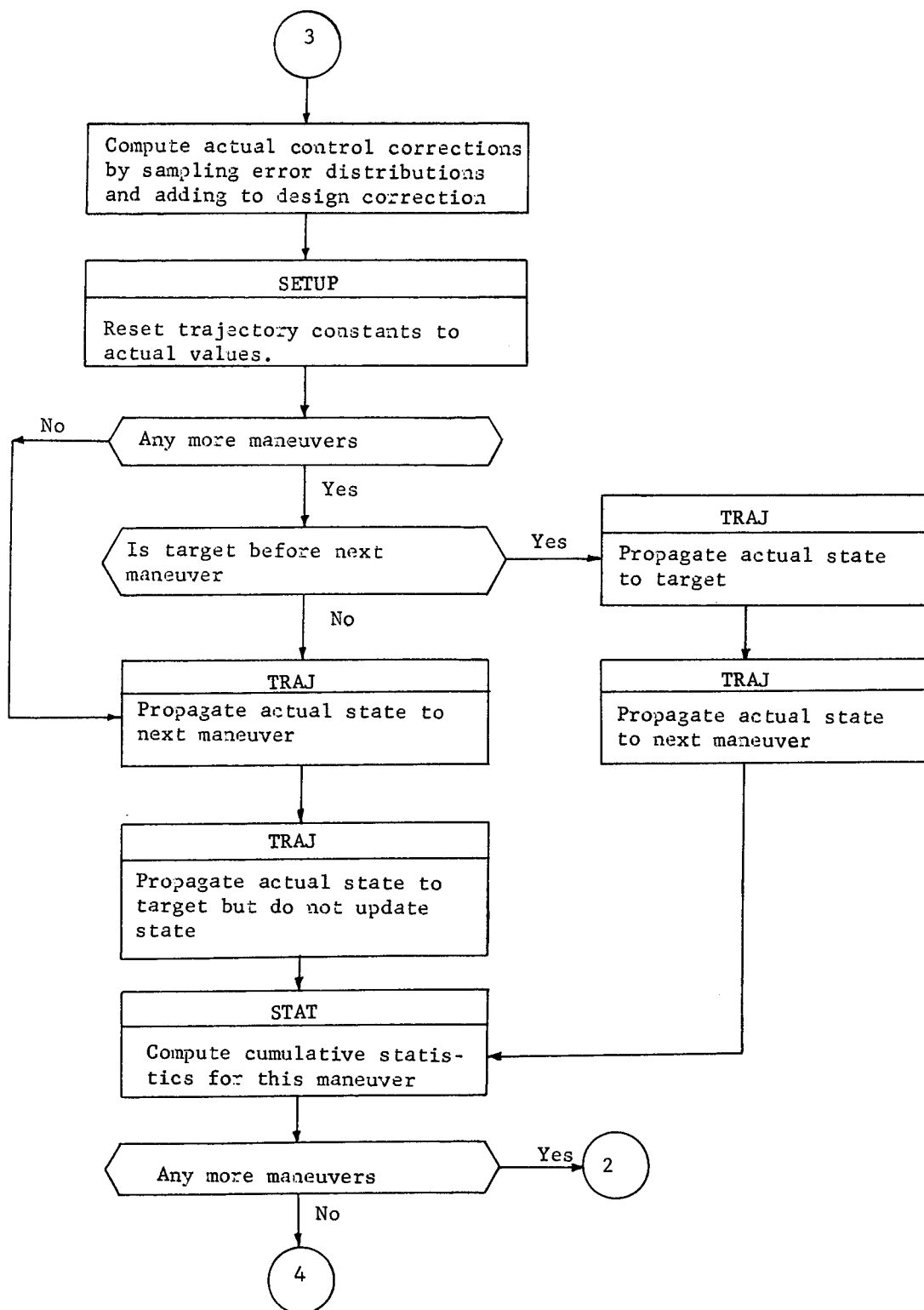
Input: see TSIM Program Description (Section 3.3)

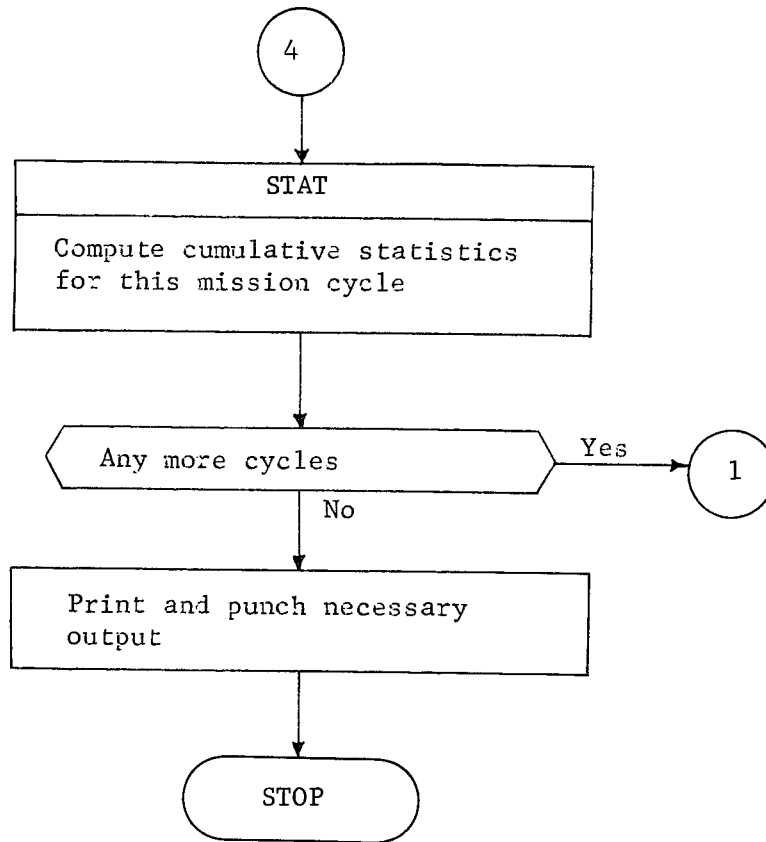
Output: o printout and punched cards

Remarks:

See TSIM Program Description (Section 3.3) and Macrologic (Section 4.1.3).







6. PROGRAMMING GUIDELINES

To facilitate the conversion of FRACAS to computer systems other than Control Data and to meet core restrictions certain programming practices should be followed:

- o Data names, labeled common names, and routine names will be restricted to six or fewer characters
- o FØRMAT statement literals will be defined as Hollerith fields
- o Alphanumeric constants will be six or fewer characters
- o Numeric literals will have eight or more significant digits to force double precision on IBM systems
- o Input and Output will be in READ and WRITE statements using logical files 5 and 6 (TAPE5, TAPE6)
- o Variably sized matrices will be treated as vectors to comply with matrix routine requirements (see Section 5.1.1)
- o Files will be defined with the minimum allowable buffer size
- o All large arrays will be located in blank common in order to maximize memory utilization since blank common overlays the area used for program loading information at load time (this is CDC peculiar)
- o A labled common, for example WØRK, should be available for local array usage to save memory
- o Variables defining maximum array size should be in data statements in each subroutine to facilitate any subsequent increase (or decrease) in large array requirements

Labeled commons will contain related data items. Some typical commons could be:

- o FLAGS - contains all logic control flags
- o ENGINE - contains all low thrust engine parameters
- o CØNTRL - contains thrust controls
- o EPHMRS - contains planetary ephemerides and other planetary data
- o INTGRT - contains all trajectory integration data
- o STATE - contains spacecraft and planetary state vectors
- o EVENT - contains all event information
- o AUGMNT - contains all state parameter augmentation data
- o TIME - contains start time, final time, current time, etc.
- o WORK - contains working-storage memory for local usage
- o CØNST - contains commonly used natural constants

7. FUTURE OPTIONS

The inclusion of future options in FRACAS is simplified as much as possible by two main features: modularity, which has been emphasized previously, and program coding standards (discussed in Section 6. Programming Guidelines). Potential options or changes to the program fall into seven categories.

- o Dynamic model - both S/C and environmental characteristics
- o Integration algorithm
- o Transition matrix generation techniques
- o Additional state parameters
- o New data types
- o New filtering algorithms
- o Guidance algorithms

Changes to the dynamic model are localized to subroutine TRAJ. Of course, a major rewrite of the dynamics - most likely for different thruster modeling - would mean extensive changes to TRAJ. However, the only other important effect would possibly be to change the state parameter list, which is discussed below.

A new integration algorithm would force replacement of INTEG and possibly a reformulation of the dynamics in TRAJ, but this is again a localized change.

Transition matrix generation is currently done in TRAJ and PTRAN, both of which are controlled by PATH. Since these already accommodate both integration of variational equations and numerical differencing, the only other possible technique is an approximate analytic technique. Its implementation would require a new subroutine and a modified calling sequence in PATH.

Additional state parameters impact transition matrix generation (TRAJ, PTRAN) and both propagation (TRAJ, PROP) and updating (TRAKM, FILTER) in filtering. If transition matrices for these parameters were to be done variationally, or if covariance integration were to be used, the relevant equations would be needed in TRAJ. However, the existing PTRAN could compute numerically differenced transition matrices. PROP also is unaffected. It needs only a set of covariances and a set of transition matrices - all of whose dimensions are already variable. Since TRAKM currently evaluates all observation sensitivities analytically, new equations would be needed. If sensitivities by numerical differencing were preferred, a differencing routine could also be added. FILTER, like PROP, would be unaffected.

One more important possible effect of additional state parameters, however, would be felt throughout the entire program. If additional parameters resulted in a required increase of the maximum dimensions of any array, such as a priori consider covariances, a program-wide dimension change would be required. This is the reason for the programming guideline which requires that any time the maximum dimensions, rather than current working dimension, of an array are used by the program, those dimensions must be defined by a variable passed through common, and not by a local constant.

New data types, once modeled, require only additional coding in TRAKM. A new filtering algorithm, assuming it is linear, could use the existing FILTER subroutine. Only the gain matrix calculations would be affected. The only exception to this would be either a batch or non-linear algorithm. The batch algorithm is of questionable validity for the low thrust problem because of process noise. A non-linear algorithm violates the basic linearity assumption of FRACAS and would, in all likelihood, require a completely new program.

Guidance algorithms are, again, local to the GUIDM secondary overlay and to GUIDS (in TSIM), and have no macro-impact.

A potential problem area common to all future options is the over-running of the 70,000₈ word core restriction. If the overrun is minor, localized maximum array dimension could be reduced to gain core. However, if the overrun is major, only two alternatives exist. Either the 70K requirement must be abandoned, or the TEAM primary overlay must be divided into two or more primary overlays, which is certain to increase execution time and could also reduce some program flexibility, particularly with respect to event types such as guidance and prediction.

8. REFERENCES

1. "Space Trajectories Error Analysis Program (STEAP): Volume I (Analytic Manual), Volume II (User's Guide), Volume III (Programmer's Manual)," E.D. Vogt, et al, MMA Document MCR-71-3, December 1971.
2. "Astronautical Guidance," R.H. Battin, McGraw-Hill, 1964.
3. "POST-Program to Optimize Shuttle Trajectories: Volume I (Formulation Manual)," D.E. Cornick, et al, MMA Document MCR-71-287, January 1972.
4. "QUICKTOP," A.C. Masey, NASA Ames Report MS-71-1, 11 June 1971.

9. APPENDICES

The following three appendices contain technical analyses in support of FRACAS program design. Each appendix is self contained with its own references. The first Appendix (9.1) discusses Error Sources for near-Earth and interplanetary missions. The major error source is due to thruster performance. Appendix 9.2 is a study of numerical accuracy of the covariance formulation. For a pre-flight error analysis program using CDC 6000 series computers, the covariance form is sufficiently accurate with no need for a square-root formulation. Appendix 9.3 evaluates the advantages of two different covariance propagation methods: mapping with transition matrices and integrating covariance matrix differential equations. Transition matrix mapping is recommended for a pre-flight program because of its computational speed.

APPENDIX

9.1 Guidance and Navigation Error Sources

INTRODUCTION

A necessary part of any mission analysis, in particular guidance and navigation studies, is the identification of all pertinent error sources. The following survey seeks to summarize those error sources which apply to near Earth and interplanetary unmanned missions. The emphasis is on missions using continuous low thrust propulsion, but results can be used in ballistic missions since they are a subset of low thrust missions.

NOMINAL ACCELERATIONS

Quite often error sources are given as some percentage of a nominal value. It thus becomes necessary to understand the relative differences among the various forces acting on a spacecraft. Figures 1 and 2 illustrate, for the interplanetary and near-Earth environment, the major accelerations affecting spacecraft motion. The range of low thrust acceleration covers about .5 lb to .01 lb thrustors. The values for radiation pressure (and drag) assume large solar arrays (area/mass $\sim 1 \text{ m}^2$)

It is observed that for low altitude Earth orbits, the low-thrust propulsion system does not overcome drag decelerations until about 400 Km altitude. Furthermore, the thrust levels for near-Earth missions are much lower relative to primary body acceleration than for interplanetary missions, which means that many revolutions about Earth would be required to raise an orbit from low Earth altitudes to geosynchronous. Since nuclear electric power decays exponentially over long times (years) and not as a function of heliocentric distance, it is quite possible to have thrust levels greater than solar gravity for outer planet missions.

I. DYNAMIC (NON-THRUST RELATED) ERRORS

Radiation pressure - Errors from 1 to 3% (1 σ) of the nominal radiation pressure are due to (1) surface degradation as the thermal environment changes, (2) inability to predict radiative/absorptive properties of all materials involved, and (3) changing effective area due to attitude motion with respect to the sun.

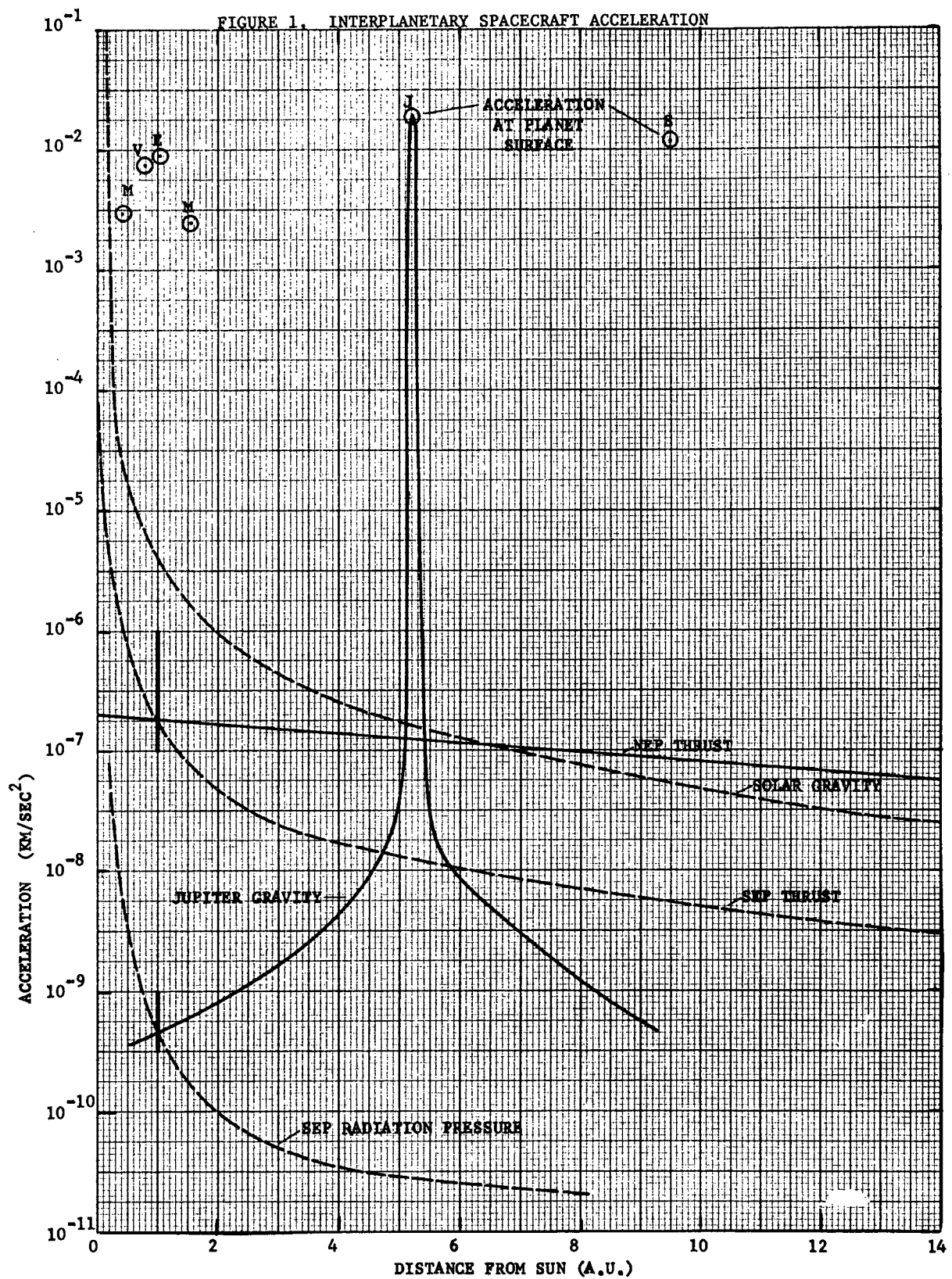
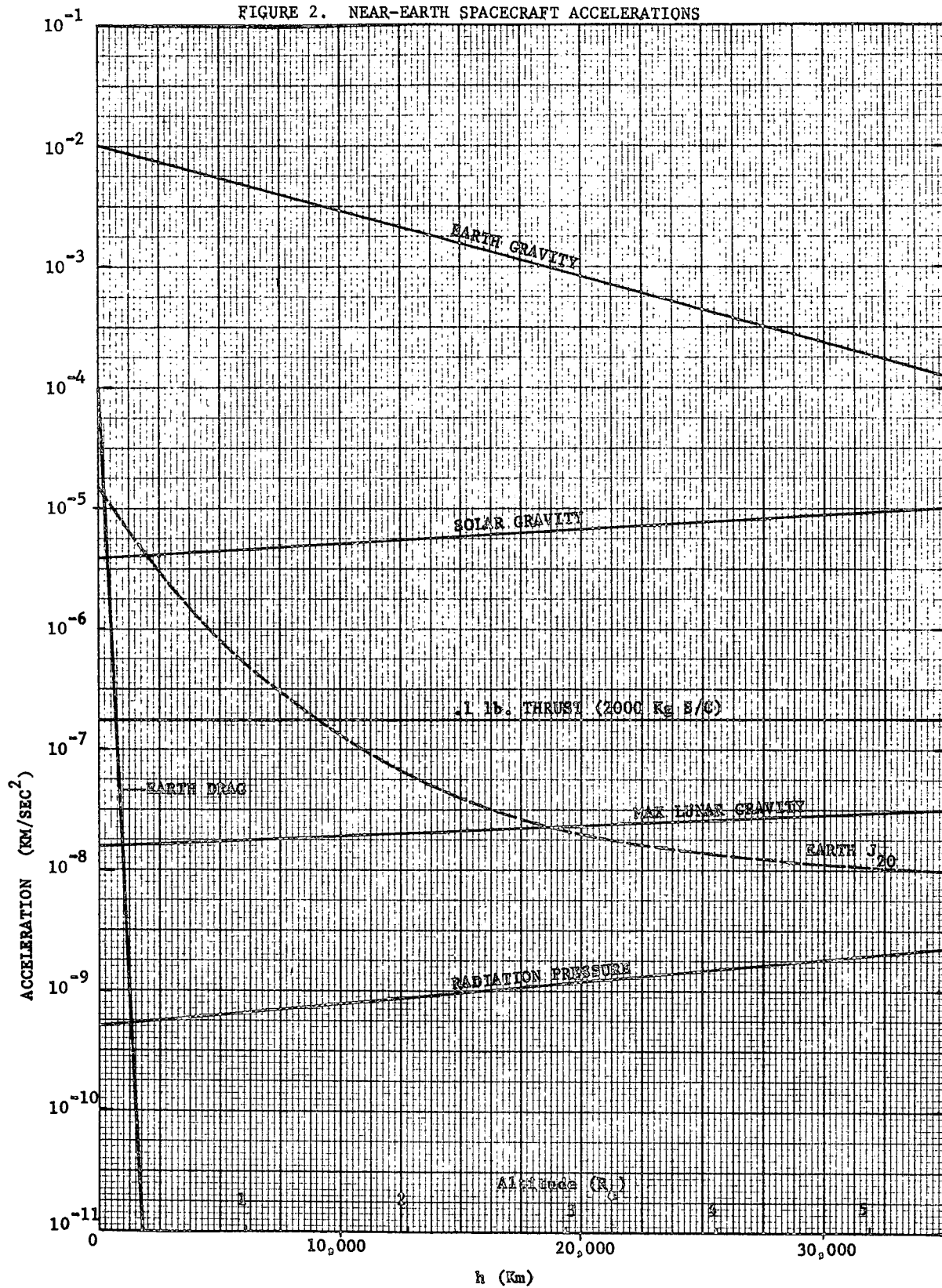


FIGURE 2. NEAR-EARTH SPACECRAFT ACCELERATIONS



Gravitational - Planetary ephemeris determination and prediction is a function of the quantity and quality of Earth based observability, using both optical and radar instruments. Accuracy is usually about 2 to 10 μ -rad (.4 to 2 arc sec) with radial (Earth line of sight) better than out of plane better than transverse (along velocity) determinations. For terrestrial planets (Mercury to Mars), RSS $\sigma \sim 100$ Km, outer planets $\sigma \sim 1000$ Km, comets and asteroids $\sigma \sim 10,000$ Km. Mass uncertainties vary widely from 2% of nominal μ for Pluto to $10^{-3}\%$ for the Sun. Earth mass uncertainties are $10^{-4}\%$ while comet and asteroid uncertainties are generally large, 1 to 50% of their nominal μ .

Venting - Semi-random accelerations can be caused by outgassing from various scientific instruments, RTG's, propulsive valve leakage, or attitude control miscoupling. These accelerations generally vary from 10^{-12} to 10^{-13} Km/sec².

Earth atmosphere - The upper atmospheric density varies with the time of day, solar cycle, and a host of other phenomena. These variations along with changing effective spacecraft area result in drag uncertainties about 1 to 10% of the nominal drag force.

Asphericity - Planetary figures are dominated by the second zonal harmonic J_2 . For Earth $J_2=10^{-3}$, moon $J_2=2 \times 10^{-4}$, Jupiter $J_2=3 \times 10^{-2}$. Except for the Earth, whose J_2 accuracy is about .01%, oblateness uncertainties for the planets and moon are about 10%.

Miscellaneous - Accelerations due to solar wind, micrometeorite impact, general relativity, etc., are usually ignored because their aggregate acceleration is less than 10^{-13} Km/sec².

Table 1 summarizes the nominal accelerations and typical uncertainties for near-Earth ($h=20,000$ Km or $3 R_e$) and interplanetary ($r=1A.U.$) regions. The dominance of one error source over another is only weakly related to their respective nominal accelerations.

SOURCE	ACCELERATION (km/sec ²)	
	Nominal	1 σ Error
Earth gravity	1×10^{-3}	1×10^{-9}
Solar gravity	6×10^{-6}	6×10^{-13}
Thrust	1×10^{-6}	3×10^{-8}
Earth J_2	3×10^{-8}	3×10^{-12}
Lunar gravity (max)	3×10^{-8}	1×10^{-12}
Radiation pressure	1×10^{-9}	2×10^{-11}
Venting	10^{-13}	10^{-13}
Miscellaneous	10^{-13}	10^{-13}
Earth atmosphere	10^{-40}	10^{-41}

TABLE 1. SPACECRAFT ACCELERATION ERRORS

II. DYNAMIC (HIGH-THRUST RELATED) ERRORS

High thrust or impulsive or chemical propulsion is characterized by small Isp (~ 300 sec) and short burn times. These propulsion systems are used for midcourse and/or orbit insertion. Pointing errors are associated with establishing and maintaining an inertial orientation during the burn. For Mariner class midcourse engines, 1σ pointing is about 7 m-rad. Proportionality errors result from propulsion parameter uncertainties and variations during the burn with $\sigma \sim 1\%$. Resolution or quantization errors are associated with cut-off sensing using timers and/or integrating accelerometers with $\sigma \sim .01$ m/sec. Generally, proportionality and pointing errors decrease as burn length increases.

III. DYNAMIC (LOW-THRUST RELATED) ERRORS

The most popular thruster by far is the electrostatic Mercury ion bombardment engine. Discussion will be confined to this thruster type although the general technique in obtaining effective thrust errors can be applied to any other thruster type. Figure 3 illustrates the typical configuration. The power conditioner moderates any power fluctuations from the solar array (or any other power source). Error sources are broken down into (1) accelerating voltage errors caused by voltage regulation, neutralizer variations, and local potential changes, (2) beam current errors are caused by mixture uncertainties among singly and doubly ionized Mercury, main vaporizer controls, and beam signal control, (3) beam spreading (with net resultant loss of thrust) is caused by distortions in electric field shape and physical grid warpage due to initial placement or on-going thermal effects, (4) pointing errors are caused by control loops for

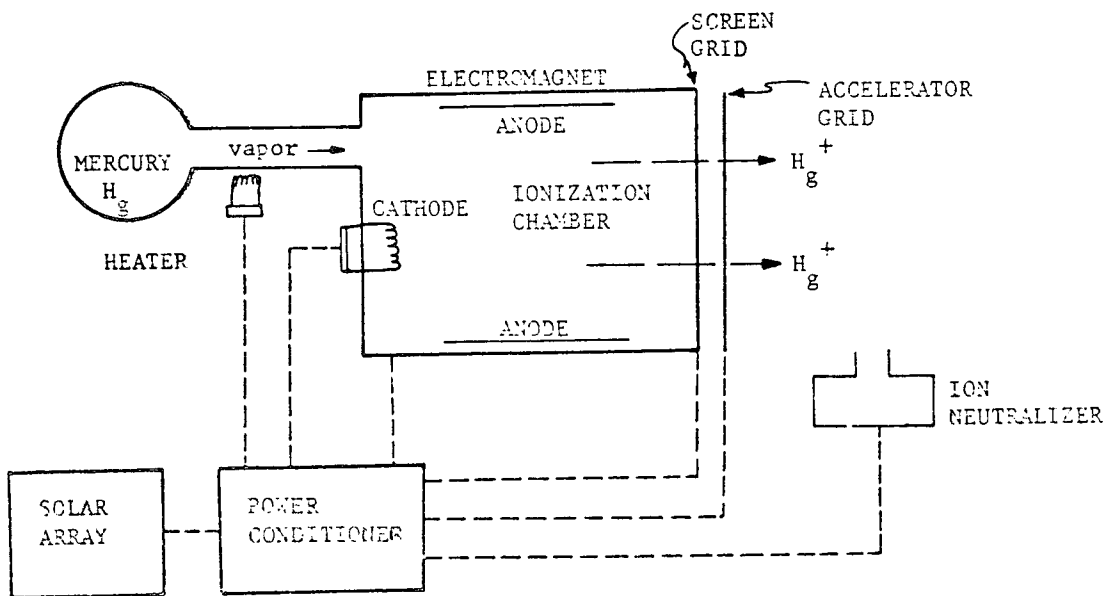


FIGURE 3. MERCURY THRUSTOR

automatically gimbaling and translating the thruster array; use of thrusters in attitude control mode reduces the thrust in the desired direction and introduces normal forces, and (5) failures can be continuous arc-outs (shorts between the grids due primarily to impurities) and outright thruster or power conditioner failure which require corrective action, either ground or spacecraft initiated; the time lapse between detection and correction may be significant if the ground is in the control loop.

Combining all of the engine errors into effective thrust errors permits a general input to guidance and navigation error analysis programs. Table 2 shows the contribution of each error source to the total effective error. If each error is assumed independent,

$$T = \frac{\eta_1 + \sqrt{2} \eta_2}{\eta_1 + 2\eta_2} \left(\frac{1}{\cos\theta} \right) I_b \sqrt{\frac{2M}{e}} \sqrt{\frac{V}{b}} (1 + \epsilon) \quad \text{where } M/e = \text{mass to charge of Mercury}$$

PARAMETER	CALIBRATION ACCURACY (%)	STEADY STATE 1σ (%)	CORRELATION TIME	ΔT/T (%)	
				BIAS	TIME-VARYING
I _b , beam current	.5	1.5	Weeks	.5	1.5
V _b , voltage	.5	1.	Weeks	.25	.5
cosθ, divergence	2.	3.	Weeks	2.	3.
η ₁ , single ion eff.	1.	5.	Days-Weeks	.02-.05	.1-.2
η ₂ , double ion eff.	20.	25.	Days-Weeks	.5-1.25	.5-1.
ε, fudge factor	30.	30.	Days-Weeks	.15	.15
Pointing	—	2 deg	?	—	3.5 cross axis

TABLE 2. MERCURY THRUSTOR ERRORS

the net bias is about 2% (1σ) and the time-varying thrust error (process noise) is about 3% and 2 deg. with correlation time about a week. The principal engine errors are beam divergence and pointing.

II. MEASUREMENT ERRORS

The primary data types for near-Earth and interplanetary missions are Earth-based, in particular range and doppler. Besides errors associated directly with the measurement, Earth based measurements are affected by station location errors. These errors include not only physical location errors but many other processes whose effect is to perturb the spacecraft/station signal such that the station location seems to be in error. These effective station location error processes include polar motion, Earth rotation rate (which affects timing by UT-ET conversion), charged particles in both space and ionosphere, tropospheric refraction, and instrument related errors: signal delay, oscillator instability and synchronization. Figure 4 illustrates the various improvements in calibrating out error processes associated with effective DSN station location longitude errors.

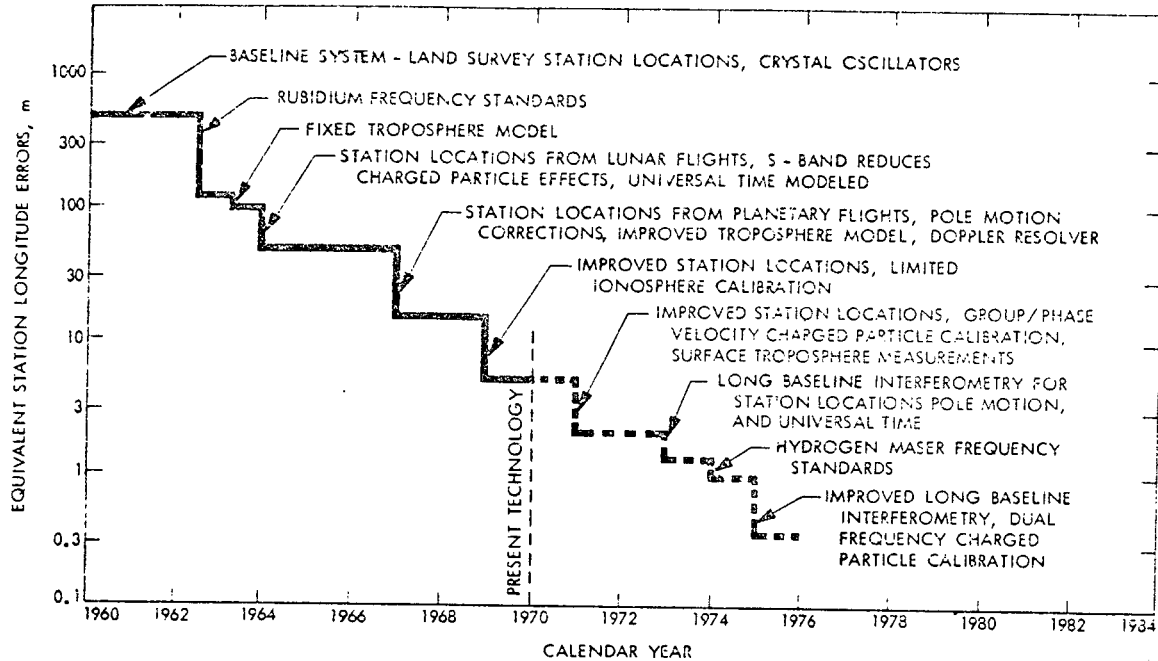


FIGURE 4. DSN STATION LOCATION ERROR IMPROVEMENT

Much of the longitude error is associated with timing errors which are common to all stations, thus longitude error is often correlated between stations at about 0.9. Although DSN location errors are on the order of 2 meters, SPADAT (Earth satellite tracking network) and MSFN (Manned Space Flight Network) have location errors about 50 meters. The higher location error present for near Earth missions is tolerated because of the shorter spacecraft to Earth distances and stronger "observability" of the Earth-based data types: range, doppler, and angles.

Current range and doppler (range-rate) uncertainties for the DSN are shown in Table 3. Typically, range measurements are weighted at a fairly pessimistic level of about 50 meters. A summary of current error levels is illustrated in Table 4 for various Earth-based measurement systems.

VHF using ground transponders and lasers are used for near-Earth tracking, as well as landmark tracking and range/range-rate from navigation satellites.

OBSERVER RELATED ERRORS	UNCERTAINTY (1σ)	
	RANGE (m)	RANGE-RATE (mm/sec)
DSN locations	3	.1
Earth rotation	2	.02
Pole motion	.7	.03
Ionosphere	—	.4
Space plasma	10	.8
Troposphere	5	.3
Station equipment	—	.4

TABLE 3. DSN RANGE, RANGE-RATE ERRORS

TRACKING SYSTEM (one r measurement/min)	NOISE (1σ)			BIAS (1σ)		
	Range (m)	\dot{r} (mm/sec)	Angles (m rad)	Range (m)	\dot{r} (mm/sec)	Angles (m rad)
DSN	50	1	—	0	0	—
MSFN/SPADAT	10	.7	.8	20	10	1.6
LASER	1.2	—	.5	.15	—	.5
VHF	15	100	—	30-100	50-200	—

TABLE 4. EARTH BASED TRACKING ERRORS

One additional Earth-based data type is the use of near-simultaneous differenced range and/or range-rate from two tracking stations. This data type, called Quasi-Very Long Baseline Interferometry (QVLBI) is proposed for future near-Earth and interplanetary missions. Expected DSN QVLBI range differencing noise is 1 to 10 meters and .1 to mm/sec for range-rate differencing. Table 5 illustrates the expected improvement in DSN QVLBI range measurements. Effective QVLBI measurements require improvements in the current DSN system, primarily in clock synchronization for range and oscillator stability for range-rate.

Error source	Present capability, m		Present configuration	Projected capability, m				Projected configuration
	Simul-taneous	Near simul-taneous		Upper value		Lower value		
				Simul-taneous	Near simul-taneous	Simul-taneous	Near simul-taneous	
Charged particles	1 ^{a, b}	1	Faraday rotation	0.1	0.5			S-X down link, 1976 Historical data improved mapping, 1973
Troposphere	1 ^{a, c, d}	1	Constant model	0.5	0.5			
Signal arrival time/ground delay	10 ^e	10 ^e	Mariner Mars 1971 planetary systems	10	10	1	1	
Clock sync	1000 ^e	1	3 μ s	1	1			Star source VLBI, 1976 H standard, 1973
Clock rate at 1 AU	3 ^f	3	Rb standard $\sim 10^{-11}$	0.3	0.3			
Transponder delay instability	0.1	1	Mariner Mars 1971	0.1	1			

TABLE 5. DSN DIFFERENCED RANGE ERRORS

A useful data type when near the target body is on-board optical data in the form of star/target body angles, target limb angles, and natural satellite (if any)/target body angles. The most efficient system makes use of the already present TV imaging instrument rather than a separate navigation device. Table 6 illustrates the optical accuracy for three systems, a current system (Mariner 9 with a 500 mm full length and 1.1x1.4 deg FOV), a projected system for outer planet missions (TOPS), and the Apollo on-board sextant.

SYSTEM	1 σ UNCERTAINTY (μ rad)		
	Noise	Bias	Distortion (constant)
Apollo	50	50	0
Mariner 9	75	25	25
TOPS	25	10	10

TABLE 6. OPTICAL NAVIGATION ACCURACY

One further instrument which is often mentioned in low thrust missions is the accelerometer. However, no accelerometers exist which can accurately measure the low thrust acceleration ($10^{-5}g$ to $10^{-7}g$) over long time spans. Both bias and scale factors are temperature dependent, about $10^{-6}g$ per deg. Fahrenheit. The experimental spacecraft, SERT II, used a "minature electrostatically suspended accelerometer" with a purported error of about 1%.

SUMMARY AND REFERENCES

A typical set of error sources for the early 1980's is illustrated in Table 7 for a Mercury orbiter SEP mission. The levels assume significant improvements over current values in almost every area.

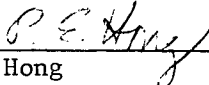
ERROR SOURCE			1 σ VALUE
Initial	RSS Position		25000 Km
S/C State	RSS Velocity		25 m/s
EP Thrust	Bias	Magnitude	0.5%
		Direction	0.5 deg
	Noise	Magnitude	2.0%
		Direction	0.5 deg
		Correlation time	1 day
Radiation Pressure			1.5%
Mercury	In-plane (ecliptic) position		5 Km
Ephemeris	Out-of-plane position		15 Km
	Gravitational constant		0.4%
DSN	Station	Radius	1 m
	Location	Longitude	.5 m
	Doppler noise (per 1 min.)		1 mm/s
	Range noise		1 m
	QVLBI	Doppler	.1 mm/s
		Range	10 m

TABLE 7. 1980 MERCURY ORBITER ERROR SOURCES

More detailed discussion of all the aforementioned error sources can be found in the references (Table 8).

ERROR SOURCE		REFERENCES
Dynamic	Radiation pressure	1,2
	Gravitational	1,2,3,5,17
	Venting	1,2
	Earth atmosphere	6,16
	Asphericity	3
	Miscellaneous dynamic	1,2
	High thrust	6,8,9,10
Measurement	Earth based (near-Earth)	4,11,12,14
	Earth based (interplanetary)	4,7,11
	Optical	14,15
	Accelerometers	10,13

TABLE 8. ERROR SOURCE REFERENCES


P.E. Hong

PEH/ac

ATTACH.

REFERENCES

1. "Mariner 4 Flight Path and its Determination From Tracking Data," G. Null, H. Gordan, D. Tito, JPL TR32-1108, Aug. 1, 1967.
2. "Mariner 5 Flight Path and its Determination From Tracking Data," G. Pease, R. Bourke, et al, JPL TR32-1363, July 1, 1969.
3. "Constraints and Related Information for Astrodynamic Calculations, 1968," W. Melbourne, et al, JPL TR32-1306, July 15, 1968.
4. "Earth Based Tracking and OD-Backbone of the Planetary Navigation System," D. Curkendall, R. Stephenson, Astronautics & Aeronautics, May 1970.
5. "Planetary Ephemerides," W. Melbourne, Astronautics & Aeronautics, May 1970.
6. "Satellite Raising to Synchronous Orbit," B. Free, Journal of British Interplanetary Society, Vol. 23, pp669-690, 1970.
7. "Applications of QVLBI Tracking Data Types to the Zero Declination and Process Noise Problems," V. Ondrasik, K. Rourke, AAS Paper 71-399, August 1971 (Astrodyn. Conf.)
8. "Solar Electric Multimission Spacecraft, Phase A Final Report, Spacecraft Subsystem Analyses," F. Goddard, et al, JPL Report 617-4, March 3, 1972.
9. Personal Communication with D. Smith, JPL, March 6, 1972.
10. "Design Considerations and Requirements for Integrating an Electric Propulsion System into the SERT II and Future Spacecraft," R. Rulis, AIAA Paper 70-1123, September 1970 (8th EP Conf.)
11. "Effects of Major Error Sources on Planetary Spacecraft Navigation Accuracies," J. Jordan, G. Madrid, G. Pease, AIAA Paper 70-1077, August 1970 (Astrodyn. Conf.)
12. "Error Studies for Ground Tracking of Synchronous Satellites," J. Cooley NASA GSFC X-551-72-30, January 1972.
13. "G&N Requirements for Unmanned Flyby & Swingby Missions to Outer Planets," D. Vraser, et al, Vol. 3: Low Thrust Missions, MIT Draper Lab R-678, May 1970.
14. "Apollo Missions and Navigation Systems Characteristics," NASA/ANWG TR-1.3, Dec. 15, 1967.
15. "Processing On-Board Optical Data for Planetary Approach Navigation," C. Acton, AIAA Paper 72-53, Jan., 1972.

16. "Atmospheric Drag Perturbations of an Earth Satellite," P. Hong, Univ. of Arizona, 1964.
17. "Accuracy of Outer-Planet Ephemerides," R. Duncombe, et al, Astronautics and Aeronautics, August 1972.

APPENDIX

9.2 Covariance Accuracy

9.2.1 Preliminary Results for Task 3 of Low Thrust OD Contract

The original intent of Task 3 was to arrive at an analytic approximation to covariance accuracy as a function of all matrix inputs to the filtering algorithm. Several obstacles arose in the analysis, so that a reasonable solution could not be obtained even for the simplest filter configuration. Consequently, a new approach is now being taken. The approach and its results will be described in a later memo.

Computational error on a digital computer results from having to express each number in the finite word length of the machine. A machine which expresses each floating point number in t binary bits can store only a t -bit approximation to the numbers desired. We therefore have two sources of error -- one from the initial t -bit approximation to the actual number, and one from the rounding or truncation when computational results must also be stored within t -bits. Since our purpose is to investigate covariance ill-conditioning, we will not dwell on such items as integrator accuracy, or the accuracy of transition or observation matrices. Since any covariance is theoretically positive semi-definite, and the mathematical operations of filtering retain this property, anytime a covariance becomes indefinite it must result from numerical problems. Since semi-definiteness is a theoretical necessity, physical subtleties such as the fact that transition and observation matrices are not exact are irrelevant to the conditioning problem. Thus, the only significant error source is the accumulated error resulting from the individual operation of addition, subtraction, multiplication and division.

The error bound associated with any single arithmetic operation on two numbers is easily determined. If the operands are exact, the resultant is accurate to within ± 1 in its least significant bit. Thus, the absolute value of the resultant relative error is less than or equal to 2^{-t} for a t -bit machine. However, if the operands themselves are in error, this result is no longer true in general. In particular, the subtraction of nearly equal quantities (or, equivalently, the addition of oppositely signed nearly equal magnitude quantities) can produce unreasonably large relative errors (see Appendix A).

Since each element in the product of an $m \times k$ matrix with a $k \times n$ matrix results from the sum of k distinct products, the potential for calculating terms with large relative error always exists for the aforementioned reasons. Thus, even for a single element in the product it is impossible to define an absolute upper bound on the relative error. However, even assuming reasonable error bounds, if the error in each matrix operation is assumed to attain its maximum value, the computed relative error bound grows to unrealistic levels (see Appendix B).

The reason this bound grows excessively is the underlying assumption that for the Euclidean norm, defined in Appendix B and denoted by $\| \cdot \|_E$, the only guaranteed bound for the norm of a product, $\|AB\|_E$, is the product of the norms, $\|A\|_E \|B\|_E$. The impact of this assumption is particularly evident in the multiplication $\bar{\Phi} P \bar{\Phi}^T$, where $\bar{\Phi}$ is the state transition matrix and P the state error covariance. For short time propagations, $\| \bar{\Phi} P \bar{\Phi}^T \|_E$ is approximately equal to $\| P \|_E$, yet $\| \bar{\Phi} \|_E$ for a particular sample Mercury orbiter mission is about 1.6×10^4 . For the Euclidean norm, $\|A\|_E = \|A^T\|_E$, so we predict a bound on the norm of $\bar{\Phi} P \bar{\Phi}^T$ to be 2.6×10^8 times the norm of P , which is ridiculously large. For a simple comparison, assume we would like to estimate the error bound only for the propagation $\bar{\Phi} P \bar{\Phi}^T$, using the same technique as in Appendix B. The straightforward analysis gives a relative error bound of 6×10^{-6} , meaning that a single propagation reduces accuracy from about fifteen digits to five or six (assuming CDC 6000 series single precision). If we make one breakdown of $\bar{\Phi}$ and P from their original dimensions of 6×6 into four sub-blocks each of dimension 3×3 , the relative error bound drops to a more reasonable estimate of losing about one decimal digit. However, the problem now becomes intractable analytically, since this same type of breakdown would be required for all consider parameters. Their magnitudes vary widely, as do the sensitivities of the state and observations to them and a special case would have to be made for each to compute meaningful error bounds.

One alternate approach would use a more optimistic error bound where we assume the error is small relative to the computed product, i.e.

$$\| \bar{E}_{AB} \|_E \approx 2^{-L} \|AB\|_E$$

This bound is suggested by Wilkinson (Ref. 2, p. 84) as acceptable in certain cases, but it can easily be exceeded and is too optimistic for the kind of assurances we need from this study. It would also be only semi-analytic in that it would require the norms of all intermediate matrices in the filtering operation, rather than depending on a few inputs as defined in Appendix B. These additional norms would have to be obtained explicitly since there are too many to evaluate analytically. The final option would be to assess the problem using statistical error bounds which is again outside the scope of this task.

REFERENCES

Wilkinson, J.H., The Algebraic Eigenvalue Problem,
Clarendon Press, Oxford, 1965

Wilkinson, J.H. Rounding Errors in Algebraic Processes, Prentice-Hall,
Englewood Cliffs, N.J., 1963

APPENDIX A

Potential Rounding Errors in Addition of Inexact Numbers

Two examples are given here of problems which can be encountered in a finite word length machine. They are designed to be illustrative. They may seem extreme - they certainly do not represent a normal situation -- but they can occur. For these examples we will assume for simplicity that our machine carries three decimal bits and it truncates rather than rounds. The first example produces an answer in error by 100% when all the input quantities are exact. The second shows an infinite relative error when the input numbers are not exact, but have been truncated after previous operations.

Example 1

Compute

$$(.601) \times (.427) + (.348) \times (-.731)$$

The result of the first multiplication is .254627 and the second is - .254388. Thus, the correct answer is $.239 \times 10^{-3}$. However, the machine will truncate each of those products to .254 and -.254 respectively, and the computed answer will be zero, yielding a 100% relative error.

Example 2

Compute $(.23125) \times (-.32) + (.121875) \times (1.92)$

The actual magnitude of each product is .234, but each multiplier which is larger than three digits must be truncated in the machine before it can be operated on. The computed results will then be

$$(.731) \times (-.32) = - .23392 \stackrel{\text{tr}}{=} - .233$$

$$(.121) \times (1.92) = .23232 \stackrel{\text{tr}}{=} .232$$

With these two results the computed sum will be $-.1 \times 10^{-2}$ which has an infinite relative error compared to the zero expected from the calculation. Similar results can be demonstrated with rounding, though the problem is slightly less severe than with truncation.

On a machine like the CDC 6000 series, where we have nearly 15 decimal digits it may take a considerable number of operations before relative errors become as large as those of the examples but they can occur in some situations. Also, relative errors much smaller than 100% can completely destroy the validity of any results.



Assume we would like to predict the state covariance error bound for a single filtering step- which consists of propagation between measurement times and an update at the latter measurement time. The resultant error depends on the initial error in the covariance; on the norms of the initial covariance, transition and process noise matrices, and the observation and measurement noise matrices; and on how each matrix enters the calculations.

Propagation from the k^{th} to the $(k+1)^{\text{th}}$ time point proceeds

$$P_{k+1/k} = \bar{\Phi}_{k+1,k} P_{k/k} \bar{\Phi}_{k+1,k}^T + Q_{k+1}$$

The measurement update consists of

$$K_{k+1} = P_{k+1/k} H_{k+1}^T [H_{k+1} P_{k+1/k} H_{k+1}^T + R_{k+1}]^{-1}$$

$$P_{k+1/k+1} = P_{k+1/k} - K_{k+1} H_{k+1} P_{k+1/k}$$

where

$P_{i/j}$ = state covariance at time t_i after processing measurements up to and including time t_j

$\bar{\Phi}_{k+1,k}$ = state transition matrix from time t_k to t_{k+1}

Q_{k+1} = process noise matrix accumulated over interval t_k to t_{k+1}

H_{k+1} = observation sensitivity matrix at time t_{k+1}

R_{k+1} = observation noise matrix

K_{k+1} = measurement gain matrix

Rewriting the filtering equations for the error bound analysis so that we have a single operation per equation and dropping unnecessary subscripts we have:

$$A_1 = \phi P_{k/k}$$

$$A_2 = A_1 \phi^T$$

$$A_3 = A_2 + Q = P_{k+1/k}$$

$$A_4 = H A_3$$

$$A_5 = A_4 H^T$$

$$A_6 = A_5 + R$$

$$A_7 = A_6^{-1}$$

$$A_8 = A_4^T A_7 = K_{k+1}$$

$$A_9 = A_8 A_4$$

$$A_{10} = A_3 - A_9 = P_{k+1/k+1}$$

From Wilkinson (Ref. 1, p. 115), we have for the error in the matrix multiplication AB

$$\|E\|_E \leq n 2^{-t}, \|A\|_E \|B\|_E$$

where

E = error matrix; difference between computed product AB and the actual product of AB

$$\|A\|_E = \text{matrix Euclidean norm defined by } \|A_{k \times l}\|_E = \left[\sum_{i=1}^k \sum_{j=1}^l |a_{ij}|^2 \right]^{1/2}$$

n = matrix dimension common to A and B

t_1 is defined from $2^{-t_1} = 1.06 \cdot 2^{-t}$ where t is the number of binary bits allotted by the computer for each floating point number's mantissa.

Similarly, for addition we can derive from equation 6.16 (Ref. 1, p. 115) that

$$\|E\|_E \leq 2^{-t} [\|A\|_E + \|B\|_E]$$

We now make the following definitions

A_i = correct matrix at the i^{th} step

A'_i = computed matrix at i^{th} step

$E_i = A_i - A'_i$ = total cumulative error at i^{th} step

$$\alpha_i = \|E_i\|_E$$

$$\beta_i = \|A_i\|_E$$

$$\beta_p = \|P_{k+1/k+1}\|_E$$

and for $Q, \Phi, H, R,$

$$\beta_Q = \|Q\|_E, \text{etc.}$$

We assume that $P_{k/k}$ is in error by E_0 . The error in $P_{k+1/k+1}$ is denoted E_{10} , and becomes the E for the next iteration. Inputs are $\beta_p, \beta_E, \beta_a, \beta_H, \beta_R$; the P matrix dimensions are $n \times n$; and the measurement is an m -vector.

The a priori error upper bounds at each step are as follows

$$\alpha_1 \leq n 2^{-z_1} \beta_Q [\beta_p + \alpha_0]$$

$$\alpha_2 \leq n 2^{-z_1} \beta_E [\beta_1 + \alpha_1]$$

$$\alpha_3 \leq 2^{1-z_1} [\beta_2 + \beta_Q] + \alpha_2$$

$$\alpha_4 \leq n 2^{-z_1} \beta_H [\beta_3 + \alpha_3]$$

$$\alpha_5 \leq n 2^{-z_1} \beta_H [\beta_4 + \alpha_4]$$

$$\alpha_6 \leq 2^{1-z_1} [\beta_5 + \beta_R] + \alpha_5$$

$$\alpha_7 \leq f(m) \cdot 2^{-z_1}$$

$$\alpha_8 \leq m 2^{-z_1} [\beta_4 + \alpha_4] [\beta_7 + \alpha_7]$$

$$\alpha_9 \leq m 2^{-z_1} [\beta_8 + \alpha_8] [\beta_4 + \alpha_4]$$

$$\alpha_{10} \leq 2^{1-z_1} [\beta_3 + \beta_Q] + \alpha_3 + \alpha_4$$

where $f(m)$ is a simple first or second order polynomial in m determined by the inversion method selected.

The resultant upper bound on α_{10} requires eight pages of algebraic manipulation to derive and one full page to write, but the key term is the multiplier of 2^{-t_1} :

$$\alpha_{10} \leq 2^{-t_1} \{ n \beta_{\bar{I}}^2 \beta_p + m \beta_H^2 \beta_7 [\beta_{\bar{I}}^2 \beta_p + \beta_Q]^2 \}$$

The relative error at the end of the ten steps is $\|E_{10}\|_E / \|P_{k+1/k+1}\|_E$ or α_{10}/β_{10} . For one sample trajectory on the approach phase of a Mercury orbiter we have the following numerical values for the relevant norms and dimensions:

$$n = 6$$

$$m = 1$$

$$\beta_{10} \approx \beta_p = 2.3 * 10^3$$

$$\beta_{\bar{I}} = 1.6 * 10^4$$

$$\beta_H = 1.0$$

$$\beta_Q = .3$$

$$\beta_7 \approx 10^5$$

Since all entries in the α_{10} computation are positive, the actual resultant upper bound can be no less than the bound computed by analyzing only the 2^{-t_1} terms. Knowing from above that $\beta_{10} \approx \beta_p$ gives

$$\begin{aligned} \alpha_{10}/\beta_{10} &\leq 2^{-t_1} \{ n \beta_{\bar{I}}^2 + m \beta_H^2 \beta_7 [\beta_{\bar{I}}^2 + \beta_Q/\beta_p] [\beta_{\bar{I}}^2 \beta_p + \beta_Q] \} \\ &= 2^{-t_1} \{ 6 \cdot (1.6 * 10^4)^2 + (1) \cdot (1.0)^2 \cdot (10^5) \cdot [(1.6 * 10^4)^2 + \frac{.3}{2.3 * 10^3}] \cdot [(1.6 * 10^4)^2 (2.3 * 10^3) + .3] \} \end{aligned}$$

Ignoring smaller terms

$$\begin{aligned}\alpha_{10}/\beta_{10} &\leq (2^{-t_1}) \cdot (10^5) \cdot (1.6 \times 10^4)^4 (2.3 \times 10^3) \\ &= (2^{-t_1}) \cdot (1.5 \times 10^{25})\end{aligned}$$

For the machines of current interest, the CDC 6000 series, $t = 48$, so

$$\begin{aligned}2^{-t_1} &= (1.06) (2^{-48}) \\ &= 3.76 \times 10^{-15}\end{aligned}$$

which gives a final relative error bound of

$$\alpha_{10}/\beta_{10} < 5 \times 10^{10}$$

Having computed an error bound which is ten orders of magnitude greater than the resultant desired matrix, we know that this particular technique cannot provide meaningful answers.

APPENDIX

9.2.2 Conclusion of Covariance Versus Square Root Filter Formulaiton Study

The conditioning effects of process noise on knowledge covariance.

The presence of process noise in the orbit determination algorithm has an important effect on resultant knowledge covariance ill-conditioning. While that presence does prevent ill-conditioning in some cases, no ill-conditioning is observed for the CDC 6000 series computers even in the absence of process noise. Therefore, the full covariance form of the Kalman-Schmidt algorithm is recommended. The square root formulation is not worth its expense.

The analytical evaluation filter accuracy is not a feasible approach, as has been discussed previously (Ref. 1). The current memo describes a new approach and its supporting computer program. Since the key to filter accuracy is computer word length, we shall investigate the filter sensitivity to word size by simulating machines of different word length. The standard internal format for floating point numbers on digital computers breaks the word into two parts, one for the most significant figures in the number's binary representations (mantissa) and the other for an exponent. This format is exactly analogous to standard decimal scientific notation. The CDC 6000 computers have a 60 binary bit word length, with 48 bits used for the mantissa. This is the longest word of any current production scientific computer and was therefore selected as the reference length for the current study. As will be shown later, covariance ill-conditioning is not a problem with this word length, which is certainly a prerequisite for its use as a reference.

The simulation program, called BANANA (Bit Allocations Necessary for Accurate Navigational Analyses), was constructed with few modifications to existing software. Word lengths shorter than the standard 48 bit mantissa are simulated by truncating bits as necessary at the least significant end of the word after each arithmetic operation. This is performed in FØRTRAN by masking expressions, which are available on the CDC. Masking expressions are expressions in which logical operations are executed on the operands bit by bit. For example, if we would like to evaluate the effect of a 24 bit mantissa - equivalent to IBM 360 single precision - we define a masking variable as 36 binary ones followed by 24 zeros. Then, after each arithmetic

operation, we perform the logical product of the resultant with the mask. The leading 36 ones in the mask preserve the 12 bits of the exponent, and the first 24 bits of the mantissa, while the trailing 24 zeros blank out the 24 least significant bits of the mantissa. Since the masking variable is defined by input, any word length may be simulated by the change of a single card.

This masking operation would be very difficult to insert in the program if all filtering and propagation operations were coded in line. However, these equations are coded in the program GØDSEP as calls to matrix operations routines. Therefore the masking expressions need only be added in these sub-routines to simulate the shorter word length operations. As was mentioned in the previous memo (Ref. 1), the purpose of this task is not to evaluate numerical errors in the integrator, or transition and observation matrix generators. Consequently, each BANANA run does not regenerate a trajectory with transition and observation matrices each time. For each trajectory study, a single GØDSEP run is made to generate and store on tape all transition and observation matrices. The observation matrices are for all possible data types which could be exercised in a given study, so that BANANA has a flexible measurement schedule. When BANANA reads the transition and observation matrices required for a given comparison run, it truncates all matrix elements, so initial accuracy of each matrix is consistent with the word length being simulated.

Given an infinite word length with which to perform all calculations, the orbit determination (OD) results will be exact (again, assuming all input matrices to be exact). Even with the grossly pessimistic error bounds discussed previously (Ref. 1), it is possible to determine a finite word length which would allow the computed solution to approach the exact solution to any specified accuracy. However, this word length would be prohibitively large. In general, as word length decreases, we will see a gradual divergence of the computed from the actual solution. Since this divergence is only important as it affects our physical interpretation of the OD results, the measure of divergence must reflect our knowledge of the physical problem. The primary criteria selected, then, are the orientation and dimensions of the position and velocity uncertainty ellipsoids as represented by the state covariance. In order to compare these quantities we compute the eigen values and eigen vectors of the position and velocity 3×3 sub-blocks of the state covariance. The relative orientations are determined by computing the angles between corresponding eigen vectors. The dimensions are compared as relative in standard deviations. Both comparisons are made to the 48 bit word length OD results. A secondary comparison is made between gain matrices. For a scalar data type, the gain matrix is a vector. Considering the partitions corresponding to position and velocity, each represents a vector. The comparison made, then, is the angle between the reference and the current gain matrix for that run.

Tests for numerical instability in a given computing problem are often made by comparing the results of double and single precision computations. A simpler but less meaningful comparison is available on the CDC 6000 series through the FØRTRAN compiler. The compiler offers an option for either

truncating or rounding the results of arithmetic operations to fit them into the 48 bit word. Rounding effectively provides an additional half bit compared to truncation, and differences can often be observed between computational sets in which all inputs and operations are identical except for the differences of rounding and truncation. The masking operation previously described makes no attempt to evaluate rounding - all computational results are assumed to be truncated. This masking does, however, allow comparison of word lengths close to, but more than one half bit away from the nominal 48. Since numerical problems result from the random, cumulative effect of losing information in the least significant bits of each number, these errors accumulate differently according to word length. But this difference is extremely small for any two word lengths, both of which are sufficiently large that neither suffers from significant accuracy loss. The comparison made here, was between 48 and 44 bit mantissae, and in all cases the 44 bit results were deemed sufficiently close to the 48 bit that both maintain acceptable accuracy levels. An example of sensitivity to numerical error was found in a comparison of two runs in a region of numerical ill-conditioning. Two runs with identical a priori and one bit difference in word length yield covariances after one day of tracking which differ by more than an order of magnitude on the diagonal.

Results:

The sample trajectory selected was the last 40 days prior to Mercury sphere of influence (SOI) encounter for a 1980 Mercury orbiter. Although different trajectories were not studied, data types were varied to evaluate the effects of observability, and the process noise level changed to evaluate its impact on OD conditioning. The primary indications resulting from this study are:

- (1) the greater the disparity in observability among state vector components, the worse the ill-conditioning problem, and
- (2) moderate low thrust process noise levels have significant stabilizing effect on both long and short term OD results.

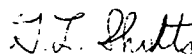
A summary of the runs made may be found in Table 1. For those runs with process noise, the nominal 1σ error levels assumed were 2% in thrust magnitude, 0.5° in thrust pointing angles and a one day correlation time.

Table 1: BANANA Results Summary

Filter Inputs Word Length	QVLBI With Process Noise	R, \dot{R} Only With Process Noise	R, \dot{R} Only No Process Noise
44 bits	SEE TEXT		
36 bits	×	×	Wild fluctuations in eigenvalues and eigenvectors
30 bits	×	×	Negative eigenvalue at 1 day
29 bits	×	×	Negative eigenvalue at 1 day
27 bits	Negative eigenvalue at 5 days	Negative eigenvalue at 5 days	×

The first and most important comparison for each case is the 44 bit to 48 bit runs. The worst case for this comparison was with conventional tracking (no QVLBI) and with no process noise. With comparative information available every two days during the arc we find that all axes of both position and velocity uncertainty ellipsoid to remain well within $.1^\circ$. Standard deviations remained well within .01% relative error with the exception of a short period of time immediately following the first ranging point. There an error in the smallest position eigenvalue did reach .2% (.1% in standard deviation) for two days. This eigenvalue was also seven orders of magnitude smaller than the other two.

The conditioning effect of process noise was best indicated by the 27 bit run with conventional tracking. After the first ranging point at five days, the smallest position eigenvalue went negative and remained negative for nearly six days. However, at the end of the tracking arc - no automatic stopping procedure was built in for negative eigenvalues - the angular differences of the position and velocity uncertainty ellipsoids from reference, were all less than one degree. All standard deviation errors were 3.5% or less. Thus, even though propagation of physically meaningless covariances occurred for over five days, the process noise level was sufficiently high to wipe out all a priori information. In other words, the latter part of the tracking arc does nothing but maintain a balance between knowledge uncertainty increases from process noise and decreases from tracking, with little effect from earlier information. We also note that ill-conditioning came later with process noise than without. In both the 29 and 30 bit runs without process noise, negative eigenvalues were observed after one day of tracking, compared to the more than five days for a shorter (27 bit) word length with process noise.



G. L. Shults

GLS/lem

Reference

"Preliminary Results for Task 3 of Low Thrust OD Contract", G.L. Shults, MMC IDC, 11 October 1972.

APPENDIX

9.3 PDOT vs. PHI

INTRODUCTION AND SUMMARY

In any linear error analysis program a major component is the propagation of state error covariances from one event to the next event. Two methods are generally used: integration of the covariance matrix differential equations (PDOT) and covariance mapping, with transition matrices (PHI). This study compares PDOT vs. PHI for low thrust trajectories from the viewpoints of both modeling accuracy and computational time. A key part of the evaluation is the process noise model which is especially critical for low thrust missions. Generally PDOT offers greater modeling flexibility and accuracy but at the cost of increased run time. It is recommended that for a pre-flight error analysis program, the PHI method and a semi-empirical noise model be used (along with certain operational guidelines) because it is 2 to 3 times faster than PDOT while retaining sufficient accuracy.

NOISE PROCESS

Given the nonlinear equations of motion

$$\dot{\underline{x}} = \underline{\dot{x}}(\underline{x}, \underline{u}, \underline{\eta}) \quad (1)$$

where \underline{x} is the spacecraft position and velocity, \underline{u} are constant dynamic parameters, and $\underline{\eta}$ are time-varying thrust parameters, these equations can be linearized about a reference trajectory such that

$$\delta \dot{\underline{x}} = f \delta \underline{x} + g \delta \underline{u} + h \delta \underline{\eta} \quad (2)$$

where f, g, h represent sensitivity partials (or transformation matrices) and $\delta \dot{\underline{x}}, \delta \underline{x}, \delta \underline{u}, \delta \underline{\eta}$ are errors in the respective dynamic parameters. Whereas eqn. 1 describes motion of the deterministic reference trajectory, eqn. 2 describes the propagation of trajectory deviations resulting from dynamic and a priori uncertainties. A linear error analysis is concerned with the propagation of state errors through the uncertain dynamic environment as affected by such events as measurement/state update and guidance (trajectory correction). Of particular interest is the behavior of the ensemble trajectory error P ,

$$P(t) = E \left[\delta \underline{x}(t) \delta \underline{x}^T(t) \right]$$

For low thrust missions the dominant error source by far is thrust error (Reference 1), both bias and time-varying. Since a good OD filter can estimate biases fairly accurately (Reference 2), the critical problem becomes the modeling of time-varying thrust error, $\delta \underline{n}$ and associated h. Desirable features of the noise process are that $\delta \underline{n}$ (1) have a zero mean, (2) be stationary in a wide sense, that is, $\delta \underline{n}(t_1)$ and $\delta \underline{n}(t_2)$ are related only by the interval $\Delta t = |t_2 - t_1|$, and (3) be time correlated such that the correlation between $\delta \underline{n}(t_1)$ and $\delta \underline{n}(t_2)$ is inversely proportional to Δt . A convenient, yet simple, mathematical model which fulfills these characteristics is the Gauss/Markov process, which for simplicity is described in the one-dimensional case,

$$\delta \dot{n}(t) = -\frac{1}{\tau} \delta n(t) + q \quad (3)$$

$$E [\delta n(t)] = 0$$

$$E [\delta n(t_1) \delta n(t_2)] = \sigma_n^2 e^{-\frac{|t_1 - t_2|}{\tau}}$$

$$E [q(t)] = 0$$

$$E [q(t_1) q(t_2)] = \frac{2}{\tau} \sigma_n^2 \delta(t_1 - t_2)$$

PDOT

Propagation of P by numerical integration of the matrix differential equations is a straightforward application of eqn. 2,

$$\dot{P} = FP + PF^T + Q$$

where P is the augmented error covariance containing the normal spacecraft state, dynamic biases and time-varying thrust errors,

$$\underline{x} = \begin{pmatrix} p \\ \underline{n} \\ \underline{u} \\ \underline{n} \end{pmatrix}$$

$$F = \begin{pmatrix} 0 & I & 0 & 0 \\ f & 0 & g & h \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -p \end{pmatrix}$$

$$Q = \begin{pmatrix} \frac{1}{\tau_1} & 0 & 0 \\ 0 & \frac{1}{\tau_2} & 0 \\ 0 & 0 & \frac{1}{\tau_3} \end{pmatrix}$$

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\rho E \begin{bmatrix} \delta\eta\delta\eta^T \end{bmatrix} \end{pmatrix}$$

This formulation has been incorporated into the low thrust error analysis program, GODSEP, by Wayne Ratliff and exercised on a SEP Mercury approach trajectory. Figure 1 illustrates the growth of spacecraft position error for various values of noise. The nominal 10 thrust error (N) is 2% in proportionality and .5 degrees in pointing and a spherical a priori state uncertainty of 10 Km in each position component and .1 m/sec in velocity. The limiting value of the noise process as correlation time (τ) approaches infinity is, of course, a bias. An important characteristic of both bias and noise is that a reduction, for example by an order of magnitude, of thrust error results in almost an exact corresponding reduction in state error growth. This is reassuring for the analyst who can then scale linearly the effects of error propagation corresponding to any given thrust error. It is also interesting to note that for the nominal error level, which corresponds to projected levels in the 1980's, the effect of noise resembles a scaled bias. This is illustrated graphically in Figure 2 and numerically in Table 1 (A through D). Indeed, an empirical formula can be derived to estimate an "effective" bias for corresponding correlated noise,

$$\sigma_{\text{BIAS}} \approx (.30 + .05 \tau) \sigma_{\text{NOISE}} \quad (\tau \text{ in days}) \quad (5)$$

One further point observed in Table 1 is the correlation of the "considered" thrust error with the state. For correlation times about one day, these correlations are quite small which indicate the relative independence of process noise with respect to state. Of course, as correlation time increases the process noise looks more and more like a bias and the correlations approach significant values.

An estimate of computer run time shows that each eigenvector event takes approximately .32 to .38 sec and each day of integration requires .15 sec when thrust noise is augmented to the basic state and .08 sec when bias is augmented (integration step size $\sim .1$ day). These values are somewhat pessimistic because the PDOT formulation is not fully optimized which particularly affects eigenvector time. A typical 42 day propagation with 3 eigenvector events takes 6.2 sec with biases and 7.6 sec with noise.

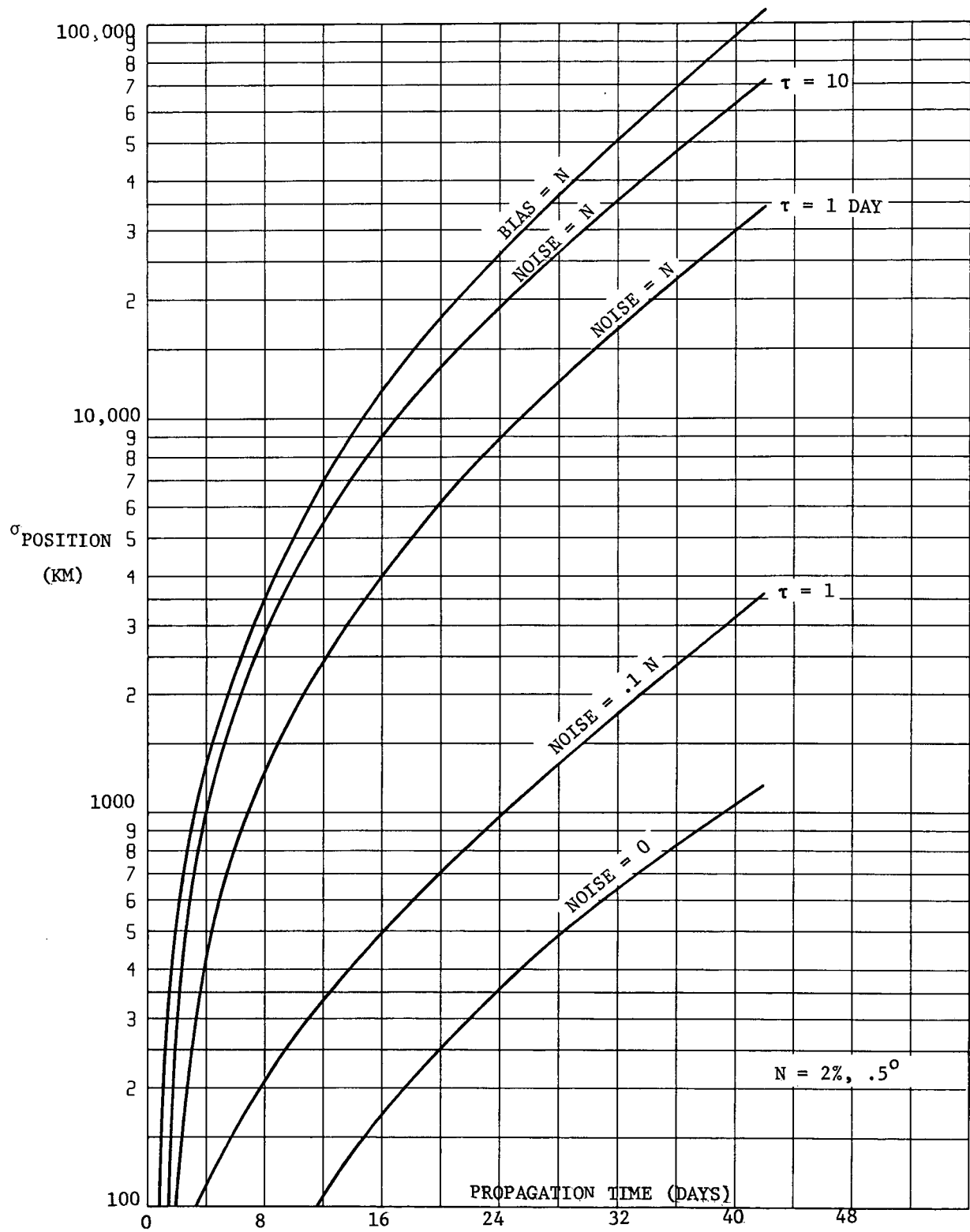


FIGURE 1. THRUST ERROR PROPAGATION

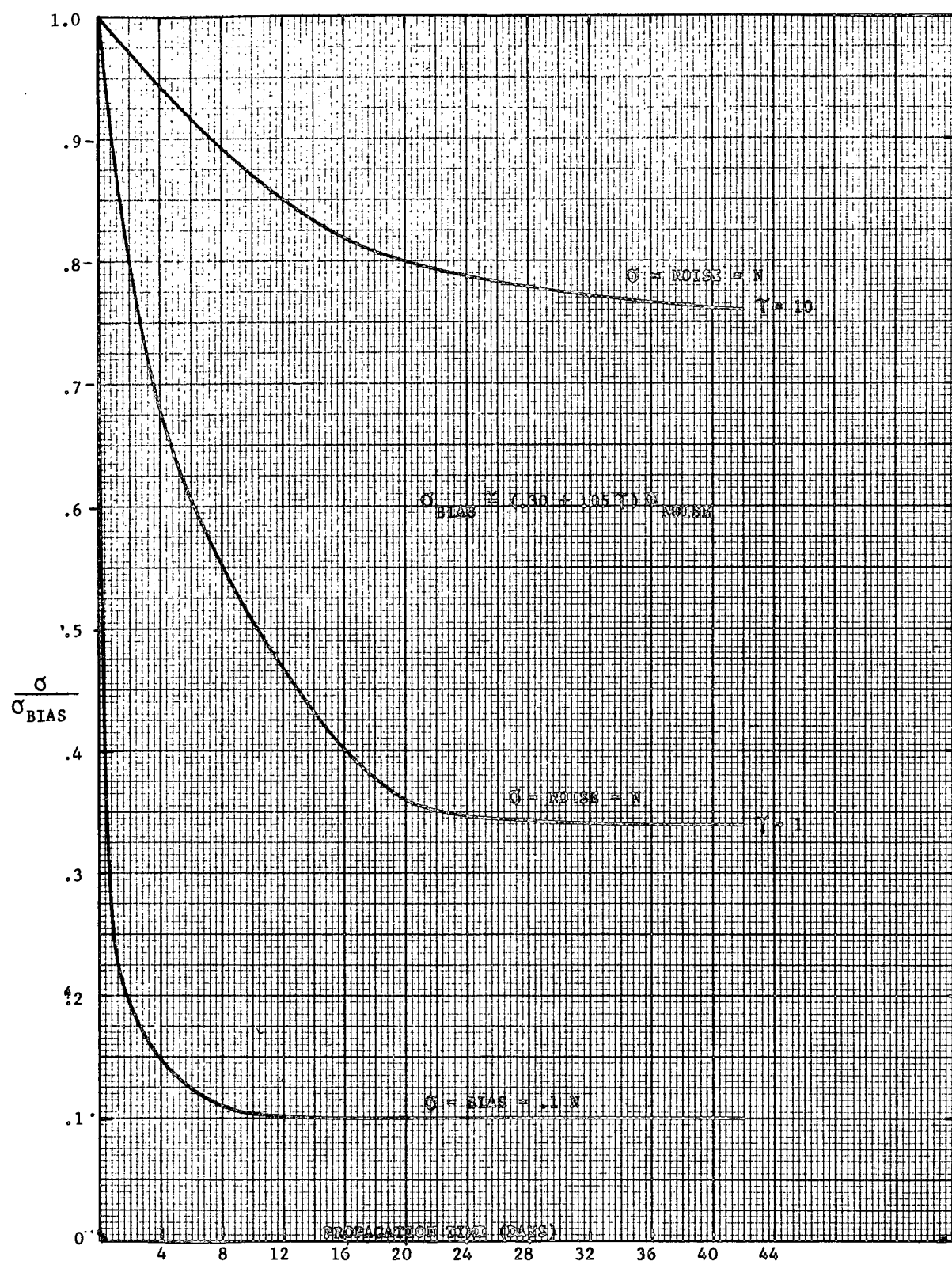


FIGURE 2. THRUST NOISE vs. BIAS

STD DEV	X	Y	Z	VX	VY	VZ
1.24468669E+04	1.00000000					
1.23469720E+04	-.60828329	1.00000000				
7.26846972E+03	-.30270308	.28646992	1.00000000			
1.41349795E-02	.99567136	-.67914560	-.31108143	1.00000000		
1.40537284E-02	-.50839711	.99268738	.27295757	-.58590746	1.00000000	
7.87201342E-03	-.32202365	.31776206	.99936949	-.33262465	.30388930	1.00000000
PROPORTIONALITY	-.93600739	.84138693	.40261172	-.96357554	.77157193	.42834695
IN-PLANE	-.34304462	-.53654712	-.00893549	-.25466830	-.63412204	-.02550814
OUT-PLANE	.07767454	-.06238571	.91509205	.08150787	-.04743576	.96322473

TABLE 1.A. STATE ERRORS AND CORRELATIONS DUE TO THRUST BIASES

STD DEV	X	Y	Z	VX	VY	VZ
9.84887652E+03	1.00000000					
1.00461282E+04	-.58714297	1.00000000				
5.75341003E+03	-.29951991	.28558389	1.00000000			
1.09290247E-02	.91612376	-.52433054	-.27375124	1.00000000		
1.15872512E-02	-.52712665	.97765142	.27782650	-.50776702	1.00000000	
5.96905027E-03	-.30588832	.28371392	.92778615	-.32576376	.28843582	1.00000000
	-.39622812	.25879867	.15473349	-.63215493	.29948109	.25425903
	-.10941933	-.29015219	-.09263148	-.10843125	-.33571167	-.00875973
	.03244149	-.01944589	.35196078	.05320495	-.01948229	.55619333

TABLE 1.B. STATE ERRORS AND CORRELATIONS DUE TO THRUST NOISE ($\tau=10$)

STD DEV	X	Y	Z	VX	VY	VZ
4.27514449E+03	1.00000000					
4.67594949E+03	-.54862167	1.00000000				
2.52478956E+03	-.29164656	.28605824	1.00000000			
4.61184034E+03	.79551156	-.33077301	-.21963966	1.00000000		
4.99703737E+03	-.53403867	.96020777	.28502762	-.39075486	1.00000000	
2.45652446E+03	-.27867356	.24440462	.82268769	-.31607338	.26731667	1.00000000
COVARIANCE						
IN-PLANE	-.01924655	.00271218	.00638302	-.20824715	.002078355	.07626072
OUT-PLANE	-.00128416	-.00773286	-.00300732	-.00968500	-.08446154	-.00009406
	.00162190	-.00022396	.01441349	.01759503	-.00170494	.17172109

TABLE 1.C. STATE ERRORS AND CORRELATION DUE TO THRUST NOISE ($\sigma=N$, $\tau=1$)

STD DEV	X	Y	Z	VX	VY	VZ
4.57308387E+02	1.00000000					
5.14918090E+02	-.42547927	1.00000000				
2.92751124E+02	-.23289450	.23050252	1.00000000			
4.67005633E+04	.78879580	-.26573676	-.18509474	1.00000000		
5.35411358E+04	-.42397601	.96380493	.23919814	-.33069923	1.00000000	
2.52545324E+04	-.24919426	.22974747	.80607646	-.30052178	.25648664	1.00000000
COVARIANCE						
IN-PLANE	-.01739334	.00246321	.00550495	-.20564315	.01939383	.07417920
OUT-PLANE	-.00120064	-.00702245	-.00000631	-.00956242	-.07882558	-.00009150
	.00151630	-.00020341	.01243070	.01737501	-.00159085	.16703415

TABLE 1.D. STATE ERRORS AND CONDITIONS DUE TO THRUST NOISE ($\sigma=.1N$, $\tau=1$)

PHI

Propagation of P by transition matrices is probably the most popular method. It relies upon transition or mapping matrices which can be generated by numerical differencing or by a procedure similar to PDOT, that is, integrating variational equations.

$$P(t) = \Phi(t, t_0) P(t_0) \Phi^T(t, t_0) + \tilde{Q}(t) \quad (6)$$

where \underline{x} contains only the basic spacecraft state and dynamic biases,

$$\underline{x} = \begin{pmatrix} \underline{x} \\ \underline{u} \end{pmatrix}$$

$$\Phi = \begin{pmatrix} \phi & \theta \\ 0 & I \end{pmatrix} \quad \text{with } \dot{\Phi} = F\Phi \text{ and } \Phi(t_0, t_0) = I$$

$$\tilde{Q} = \int_{t_0}^t \int_{t_0}^t \phi(t, s_1) h(s_1) E \left[\delta \underline{n}(s_1) \delta \underline{n}^T(s_2) \right] h^T(s_2) \phi^T(t, s_2) ds_1 ds_2$$

It is apparent that this method should be much faster than PDOT if only because of the smaller dimensional state. However, an explicit assumption is that the thrust noise $\delta \underline{n}$ and state error $\delta \underline{x}$ are independent. As we have seen in the PDOT results this is not always true, particularly for long correlation times. A further drawback is the need to evaluate the double integral for Q which would require substantial computer time unless reasonable approximations can be made.

The program GODSEP propagates P by the PHI method. The state transition matrix ϕ , is obtained by integrating variational equations and the dynamic transition matrix θ , is obtained by numerical differencing. GODSEP also contains a \tilde{Q} approximation (see also Ref. 3),

$$\tilde{Q} = R(\Delta t, \tau) \left[\alpha H(t) + \phi(t, t_0) H(t_0) \phi^T(t, t_0) \right] \quad (7)$$

where $\Delta t = t - t_0$

$$H(t) = \begin{pmatrix} 0 & 0 \\ 0 & h E \left[\delta \underline{n}(t) \delta \underline{n}^T(t) \right] h^T \end{pmatrix}$$

$$R(\Delta t, \tau) = \frac{1}{2}\tau \Delta t$$

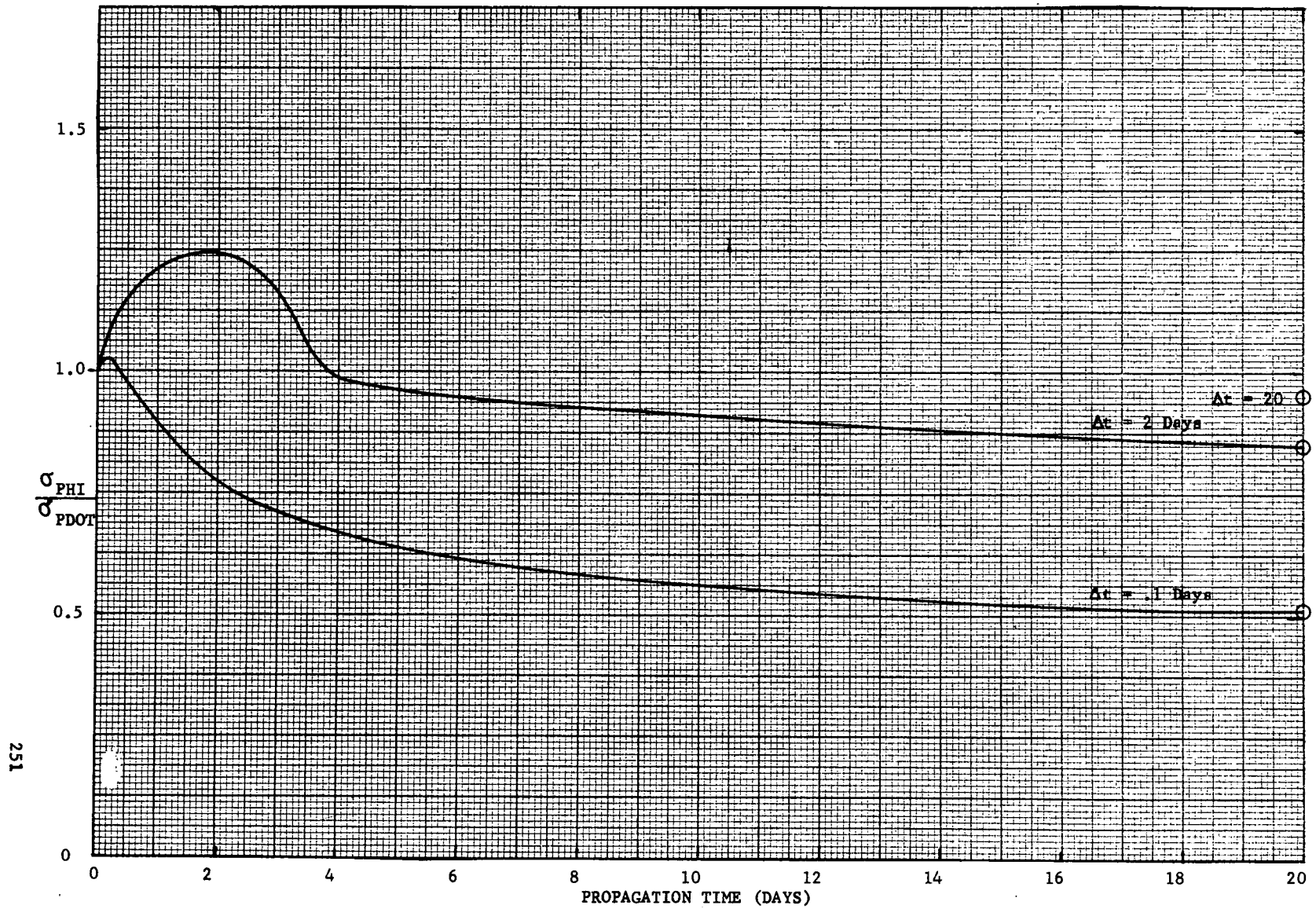
$$\alpha \begin{cases} = 2 & \text{for } \Delta t > \tau \\ = 0 & \text{for } \Delta t < \tau \end{cases}$$

It is important to note that \tilde{Q} is computed only from the last event to the current event and not at each integration step. This semi-empirical model for \tilde{Q} essentially translates thrust noise into an effective ΔV covariance and projects the covariance to the current time. For short intervals ($\Delta t < \tau$) \tilde{Q} resembles a bias.

To determine the accuracy of the PHI method (and \tilde{Q}) a 20 day propagation is compared with the corresponding integrated values for the same Mercury approach trajectory used in PDOT ($\tau=1$ day). The overall propagation time is divided into smaller intervals (Δt) to examine their cumulative effect at the end of 20 days. Figure 3 shows the results for $\Delta t=20, 2$, and $.1$ days. It is apparent that PHI propagation accuracy is good for large intervals, but breaks down for small intervals ($\Delta t < \tau$). One characteristic which is difficult to observe in Figure 3, except for $\Delta t=2$, is the pessimistic estimate occurring early and an optimistic error later.

The results of Figure 3 must be interpreted with respect to program usage. Guidance and prediction events generally require at least 10 day propagations which is comforting from an accuracy viewpoint. Measurement events fall in the other extreme of less than $.1$ day propagation. However, because each measurement alters the covariance, the cumulative effect of the combined measurement/propagation process must be considered. Figure 4 examines the effect of taking 10 days of measurements starting at the end of a 20 day propagation (to build up the a priori covariance at the start of tracking). The measurements represent a typical tracking schedule including range, range-rate, and differenced range and range-rate (QVLBI). When estimation uncertainties are compared in Figure 4 it is seen that the behavior is similar between PHI and PDOT although the "plateaus" are different. Some of the differences may be attributed to the different a priori covariances at the start of tracking. The discontinuity in estimation error at 25 days is caused by a somewhat optimistic ranging point. Table 2 summarizes the results after 10 days of tracking. The state vs. noise correlations remain small enough such that the knowledge error with PHI and Q is sufficiently close to that of PDOT. The approximate noise model of Eqn. 7 was arrived at semi-empirically and was found to be best overall. It obviously is far from perfect. Results of other models are displayed in Table 3 which compares the eigenvectors and eigenvalues at the end of tracking. Position error is more accurately predicted than velocity for all the tested models.

FIGURE 3. \tilde{Q} PROPAGATION ACCURACY



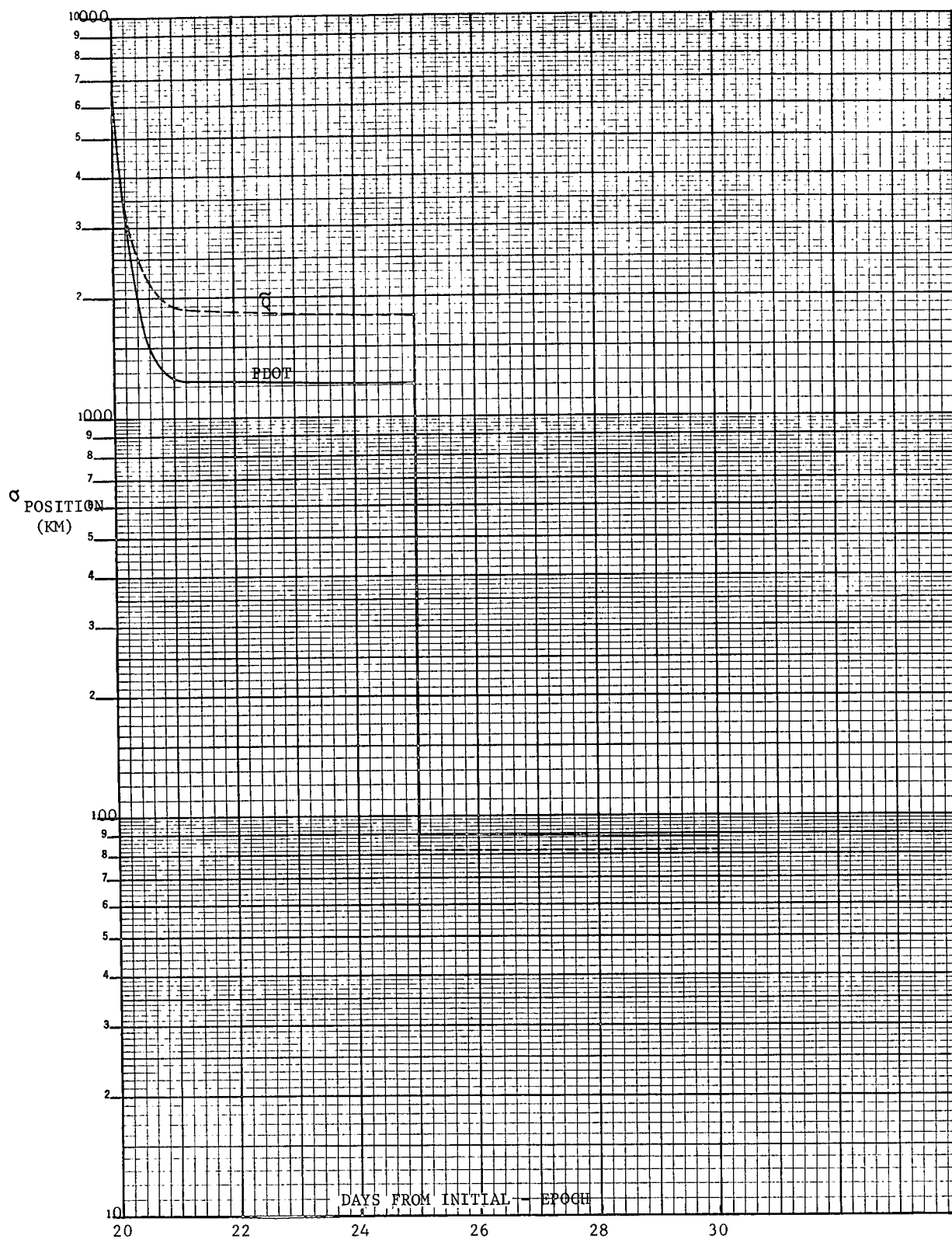


FIGURE 4. RSS POSITION ESTIMATION ERROR

STD DEV	X	Y	Z	VX	VY	VZ
2.59946888E+01	1.00000000					
7.73700156E+00	-.70346918	1.00000000				
8.01983121E+01	-.56666376	.97225223	1.00000000			
5.00631305E-04	.65280682	-.17616843	-.06196110	1.00000000		
1.23188394E-04	-.31459821	.43392646	.30723434	-.13384830	1.00000000	
6.34892223E-04	-.41222144	.69359626	.71815400	-.15540390	.41508028	1.00000000
PROPORTIONALITY	-.17346095	-.04914312	-.00222550	-.54335037	-.50978413	.07895324
IN - PLANE	.16260654	-.08997140	.01746565	.36951855	-.64228689	-.02060047
OUT - PLANE	-.00781447	.12923226	.15002027	.06900239	.19450868	.48915242

TABLE 2.A. PDOT STATE ERRORS AFTER 10 DAYS OF TRACKING

STD DEV	X	Y	Z	VX		
2.41750335E+01	1.00000000					
7.93414793E+00	-.57928657	1.00000000				
7.17839280E+01	-.54539877	.84649136	1.00000000			
4.63248612E-04	.60294047	-.04934699	-.04359343	1.00000000		
1.83811571E-04	-.08290770	.41165728	.14442886	.22293308	1.00000000	
4.55371113E-04	-.34004620	.57454667	.69320772	-.04760343	.19174122	1.00000000



TABLE 2.B. PHI STATE ERRORS AFTER 10 DAYS OF TRACKING

GODSEP →

\tilde{Q} Model for $\Delta t < \tau$		Position					
(see Eqn 7)		Eigenvalue (Km)			Largest Eigenvector Component		
R	α	x	y	z	x	y	z
$(\Delta t)^2$	0	16.7	1.9	62.4	.977	.995	.975
$(\Delta t)^2$	2	19.0	4.3	70.2	.978	.995	.976
$\frac{1}{2}\tau\Delta t$	0	19.9	4.1	73.4	.979	.995	.977
$\frac{1}{2}\tau\Delta t$	1	21.8	7.8	78.6	.981	.995	.978
PDOT		21.1	1.1	82.0	.979	.994	.977

GODSEP →

\tilde{Q}		Velocity					
(see Eqn 7)		Eigenvalue (m/sec)			Largest Eigenvector Component		
R	α	x	y	z	x	y	z
$(\Delta t)^2$	0	.249	.079	.274	.822	.995	.821
$(\Delta t)^2$	2	.414	.136	.392	.835	.994	.830
$\frac{1}{2}\tau\Delta t$	0	.471	.174	.451	.842	.990	.837
$\frac{1}{2}\tau\Delta t$	1	.641	.243	.588	.992	.988	.996
PDOT		.486	.111	.648	.959	.997	.956

TABLE 3. NOISE MODEL (\tilde{Q}) COMPARISON AFTER 10 DAYS OF TRACKING

An operational consideration in favor of the PHI method is the availability of sensitivity matrices, ϕ and θ , for output, which provide the analyst with a great deal of information on the trajectory and error processes. The ϕ and θ matrices between each event can also be stored on tape to facilitate later parametric error analyses (as opposed to PDOT which must store the F matrix at least once per integration step).

As far as computational time is concerned, PHI requires about .27 to .38 sec per eigenvector event and .04/.12 sec per day of integration without/with thrust biases. A typical 42 day propagation with 3 eigenvector events takes 2.5 sec, and 8.0 with biases. 10 days of tracking (91 measurements) consumes about 30.9 sec printing every measurement and 13.5 sec printing every fifth measurement. This compares with 25.7 sec (printing every fifth measurement) for PDOT with noise.

CONCLUSIONS AND RECOMMENDATIONS

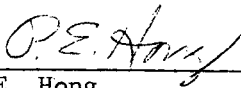
- o For correlation times of the order of 1 day, the PHI method is about 300% faster than PDOT. Sufficient accuracy is retained because of the small state vs. thrust noise correlations;
- o for correlation times about 10 days, the PHI method is about 100% faster than PDOT if noise is simulated by effective bias (Eqn.5);
- o numerical differencing for the dynamic transition matrix requires about 50% more time than integrating the variational equations. However, numerical differencing is straightforward to employ and does not require analytical partials (required in the F matrix);
- o a great deal of time is spent in print routines, particularly eigenvector and measurement, because the integration interval must be reduced to correspond to the current event and a considerable amount of data manipulation is needed to display the information properly;
- o the need for high accuracy noise modeling in \tilde{Q} at small propagation intervals is diminished by measurement processing effects;
- o operational usage favors the PHI method.

For the above reasons it is recommended that a pre-flight error analysis program propagate covariances by transition matrices, with \tilde{Q} and biases to simulate thrust noise. ϕ should be generated by integrating variational equations. As desirable options, PDOT and numerical differencing for ϕ should be available.

REFERENCES

1. "Guidance and Navigation Error Sources, " P.E. Hong, MMC IDC, 8 Sept. 1972.

2. "Guidance and Navigation Techniques for a Solar Electric Mercury Orbiter," P.E. Hong and G.L. Shults, AIAA Paper 72-917, 11 Sept. 1972.
3. "Guidance and Navigation Concepts for Solar Electric Spacecraft," P.E. Hong, MMC IDC 1643-71-23-OT, 20 May 1971.



P.E. Hong

PEH/ac